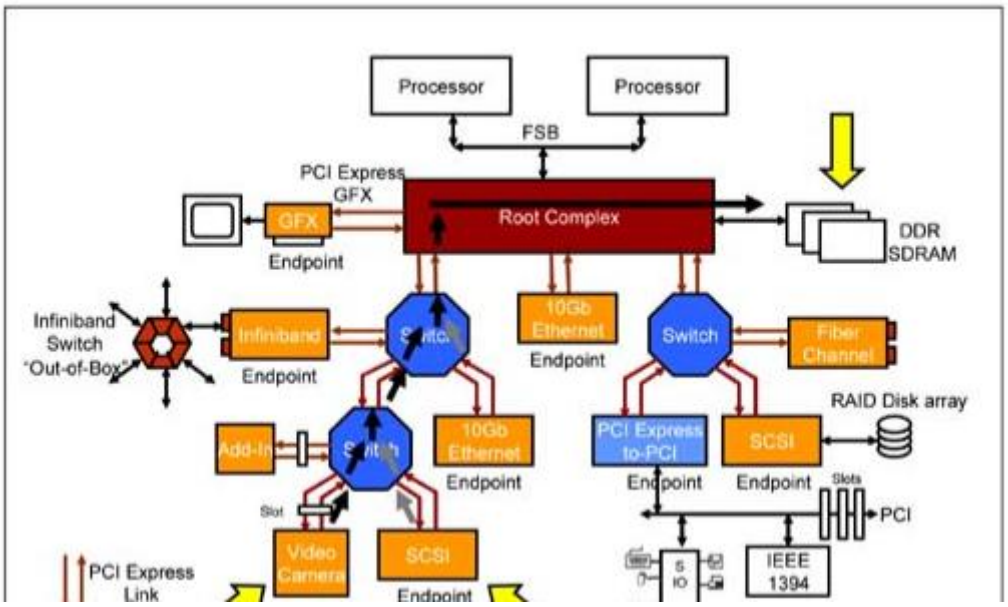
PCI and PCIe Protocol

Prepared by: Mitesh Khadgi

PCI/PCIe Device Type and Topology:



PCI bus device components

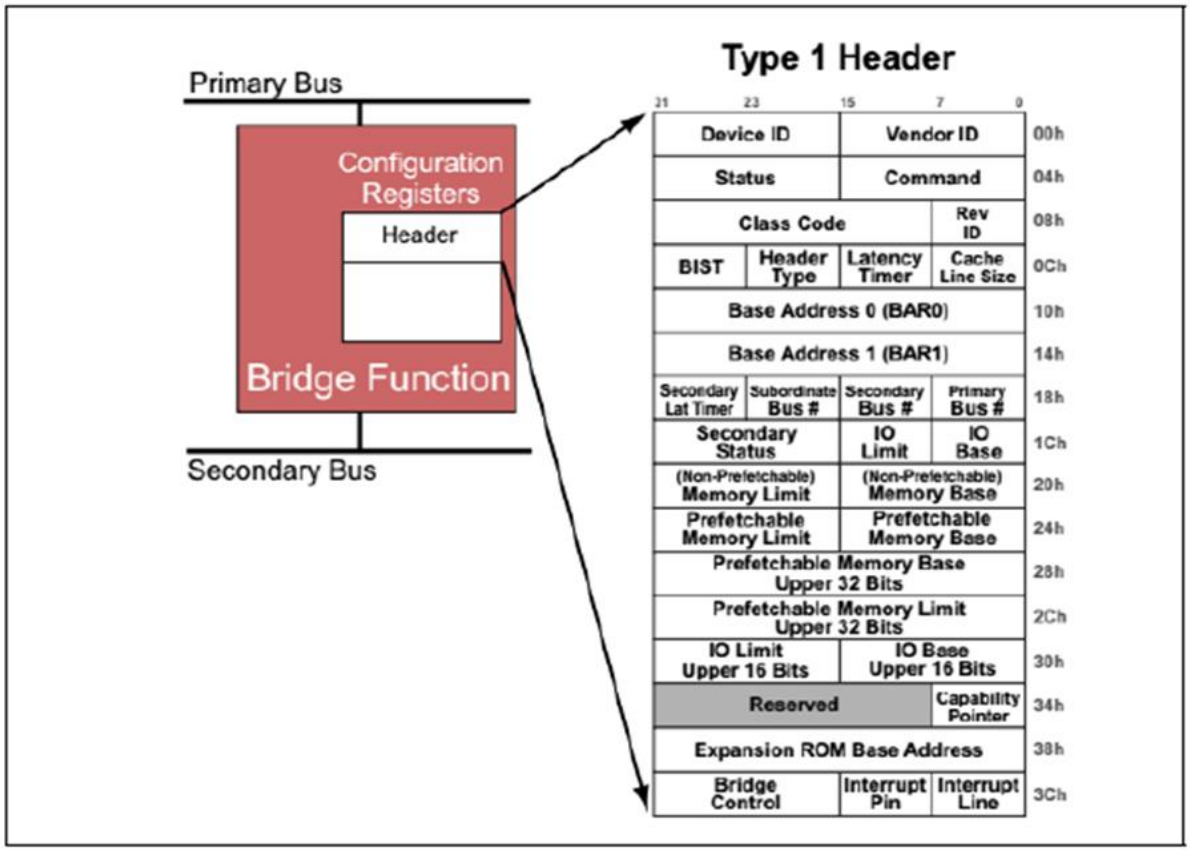
- Host bridge
- PCI bridge
- PCI device

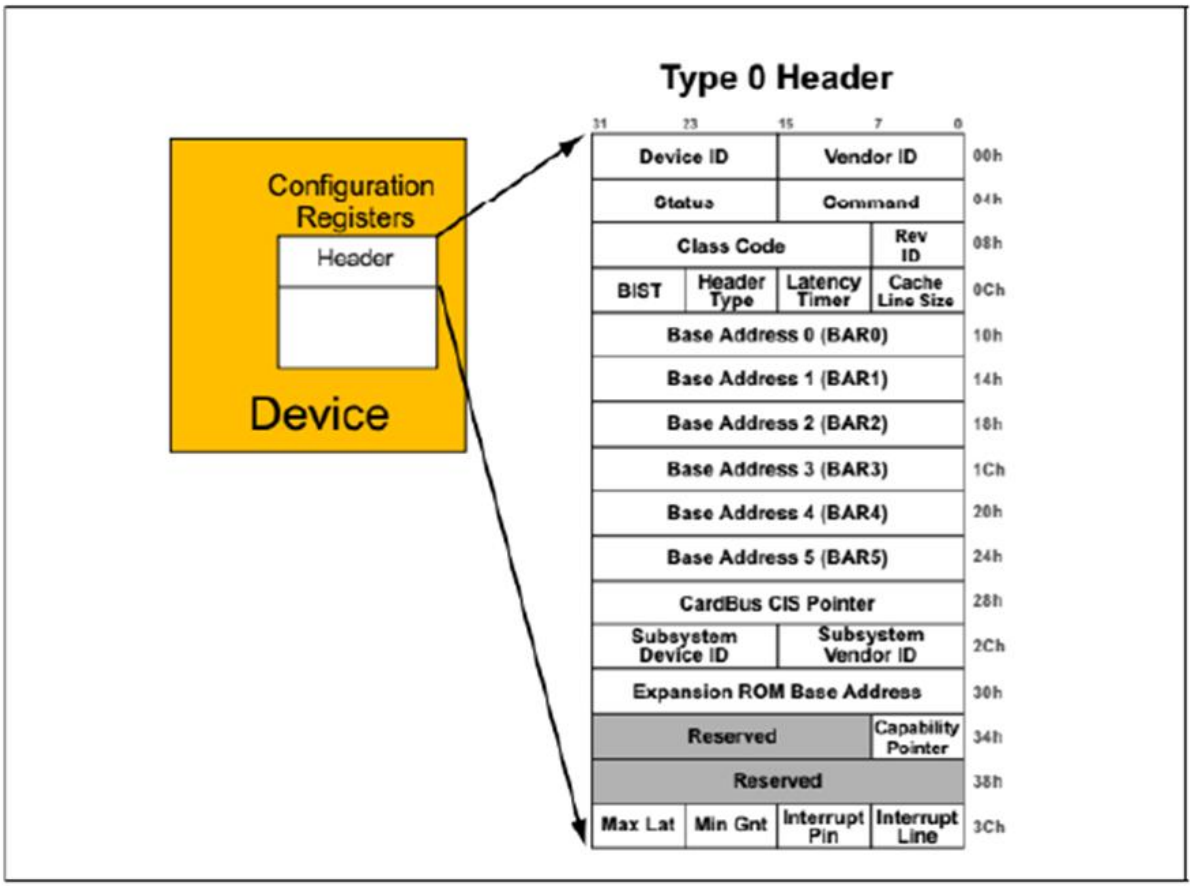
PCIe bus device components

- Root Complex (RC)
- PCIe Switch
- Endpoint(EP)

PCI device header type

- Type 1 for PCI bridge
- Type 0 for PCI device


Type 1 Header
Primary Bus
Configuration Registers
Header
Bridge Function
Secondary Bus
31 23 15 7 0
Device ID
Vendor ID
00h
Status
Command
04h
Class Code
Rev ID
08h
BIST
Header Type
Latency Timer
Cache Line Size
0Ch
Base Address 0 (BAR0)
10h
Base Address 1 (BAR1)
14h
Secondary Lat Timer
Subordinate Bus #
Secondary Bus #
Primary Bus #
18h
Secondary Status
IO Limit
IO Base
1Ch
(Non-Prefetchable) Memory Limit
(Non-Prefetchable) Memory Base
20h
Prefetchable Memory Limit
Prefetchable Memory Base
24h
Prefetchable Memory Base Upper 32 Bits
28h
Prefetchable Memory Limit Upper 32 Bits
2Ch
IO Limit Upper 16 Bits
IO Base Upper 16 Bits
30h
Reserved
Capability Pointer
34h
Expansion ROM Base Address
38h
Bridge Control
Interrupt Pin
Interrupt Line
3Ch


Type 0 Header
Configuration Registers
Header
Device
31 23 15 7 0
Device ID
Vendor ID
00h
Status
Command
04h
Class Code
Rev ID
08h
BIST
Header Type
Latency Timer
Cache Line Size
0Ch
Base Address 0 (BAR0)
10h
Base Address 1 (BAR1)
14h
Base Address 2 (BAR2)
18h
Base Address 3 (BAR3)
1Ch
Base Address 4 (BAR4)
20h
Base Address 5 (BAR5)
24h
CardBus CIS Pointer
28h
Subsystem Device ID
Subsystem Vendor ID
2Ch
Expansion ROM Base Address
30h
Reserved
Capability Pointer
34h
Reserved
38h
Max Lat
Min Gnt
Interrupt Pin
Interrupt Line
3Ch

A typical PCIe bus topology with the internal logic of RC and PCIe Switch.

RC

· Host Bridge

· Root port (Type 1 header)

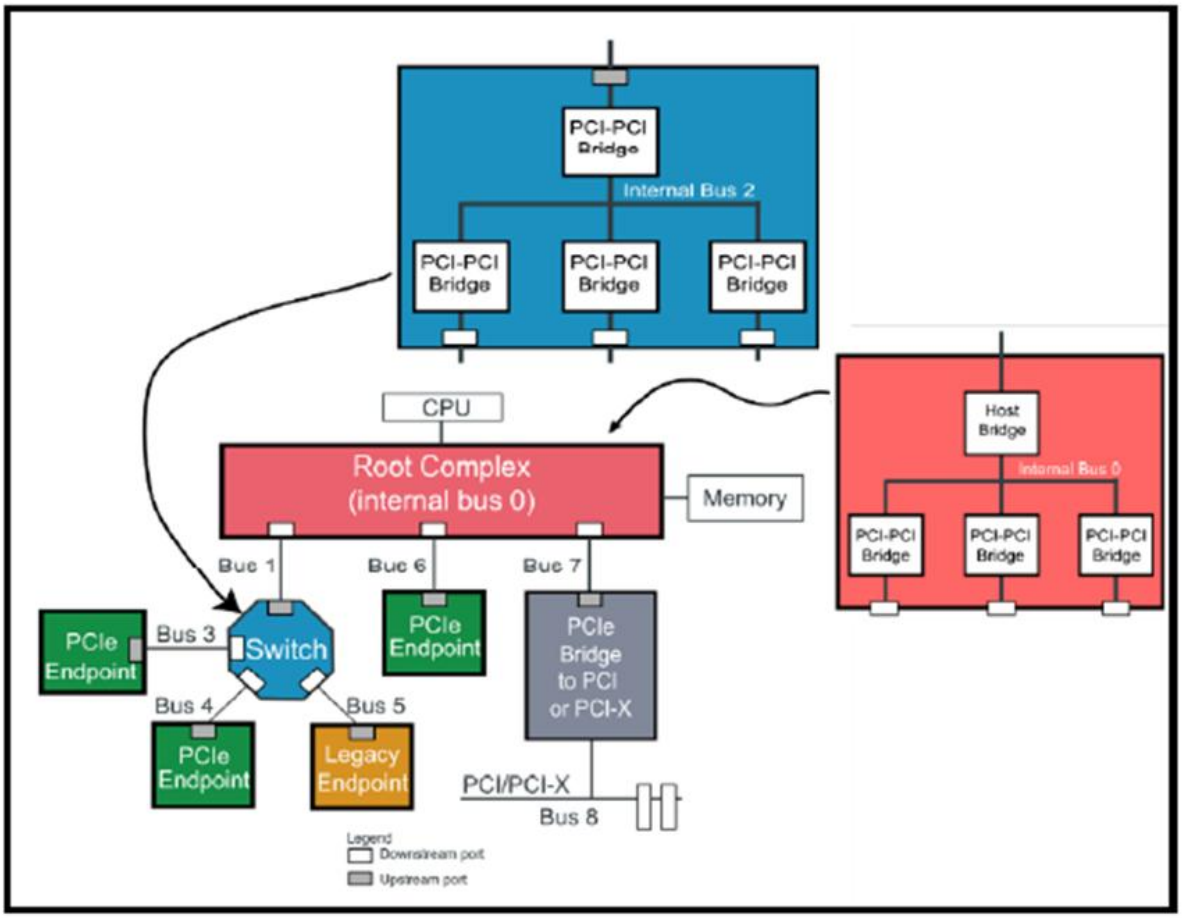
PCIe Switch

· 1 Upstream port (Type 1 header)

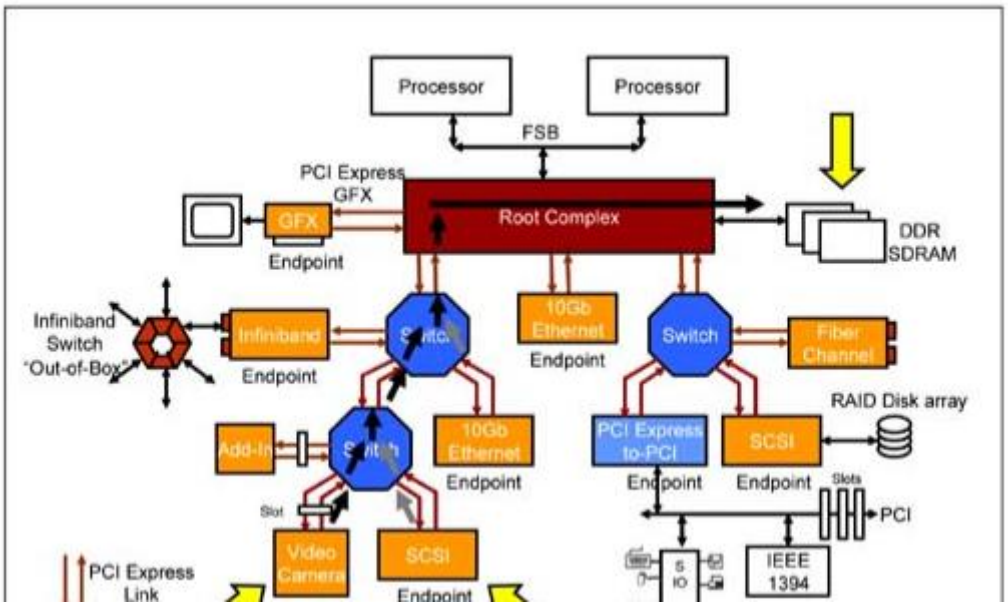
· N Downstream ports (Type 1 header)

PCIe Endpoint

· Upstream port (Type 0 header)



PCIe System Architecture:



Protocol Layers overview is as the following

- Transaction Layer

− TLP: End-to-End

− Flow Control

− QoS

− Ordering

· Data Link Layer

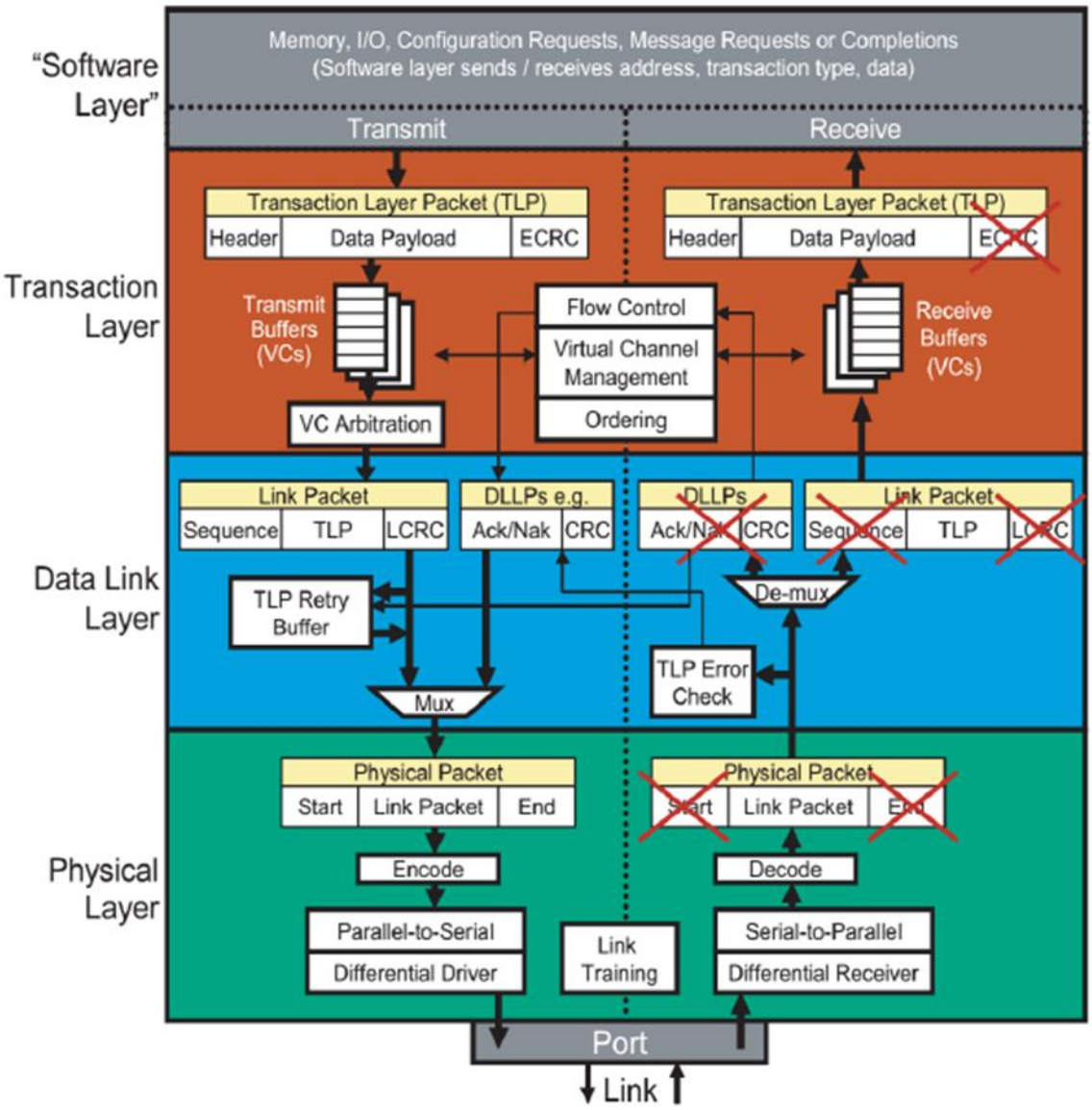
− DLLP: Point-to-Point

− Link Power management

− TLP error correction

· Physical Layer

− Ordered Set: Link training and initialization



2.1 Transaction Layer

Transaction Packet Types:

− Memory Space access

− IO Space access

− Configuration Space access

− Message

Split Transaction:

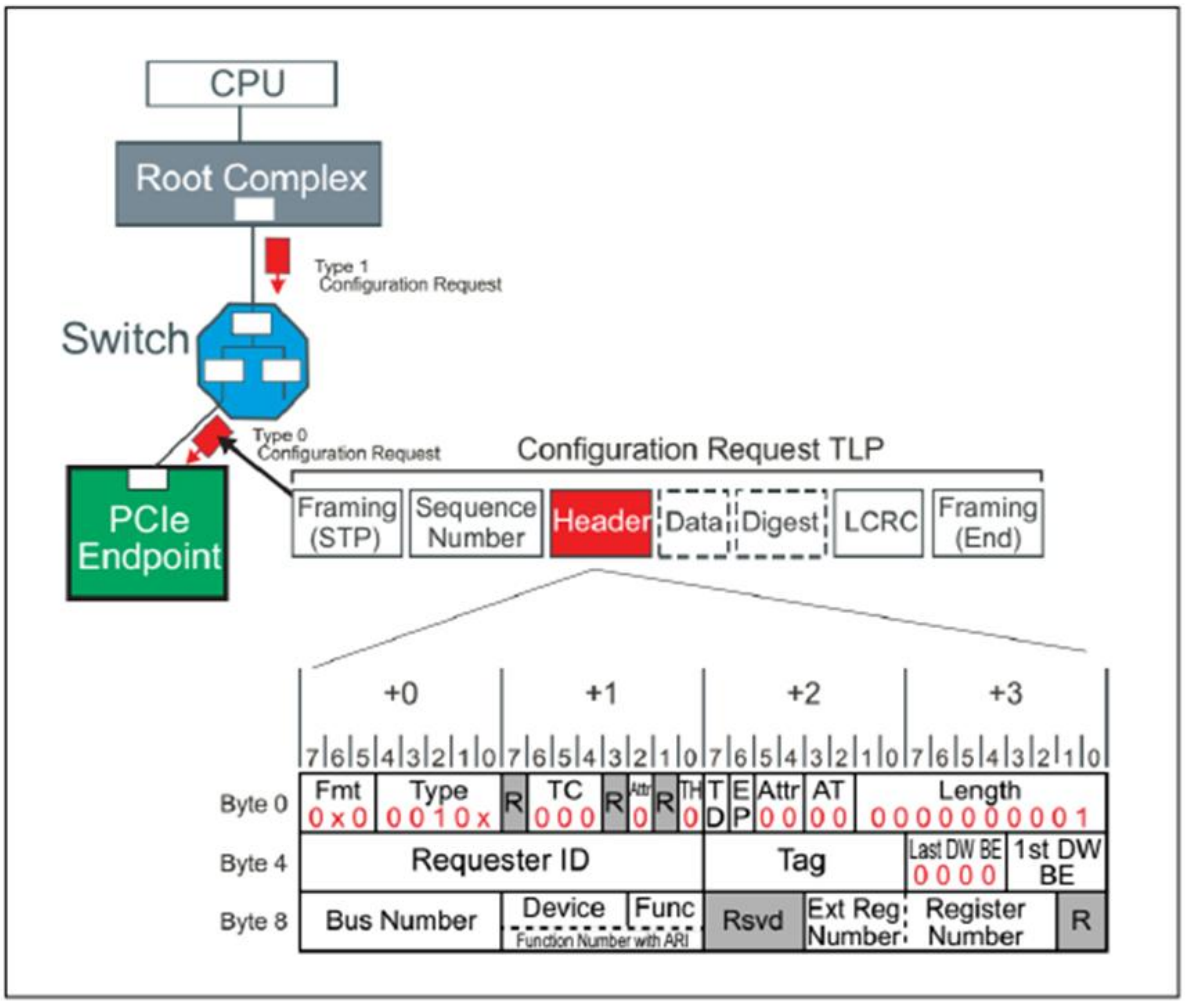
− Non-posted and Completion

Transaction Packet Types	Posted/Non-posted
Memory read	Non-posted
Memory write	Posted
IO read	Non-posted
IO write	Non-posted
Configuration read (type 0 and type 1)	Non-posted
Configuration write (type 0 and type 1)	Non-posted
Message request without data	Posted
Message request with data	Posted
Completion without data	-
Completion with data	-

TLP Header Format

- Configuration Request
- Memory Request
- IO Request
- Completion
- Message

Figure 5-9: 3DW Configuration Request And Header Format



TLP Routing

There are 3 TLP routing methods:

· ID Routing

ID routing is based on the BDF fields of TLPs.

BDF - Bus Number, Device Number, Function Number

· Address Routing

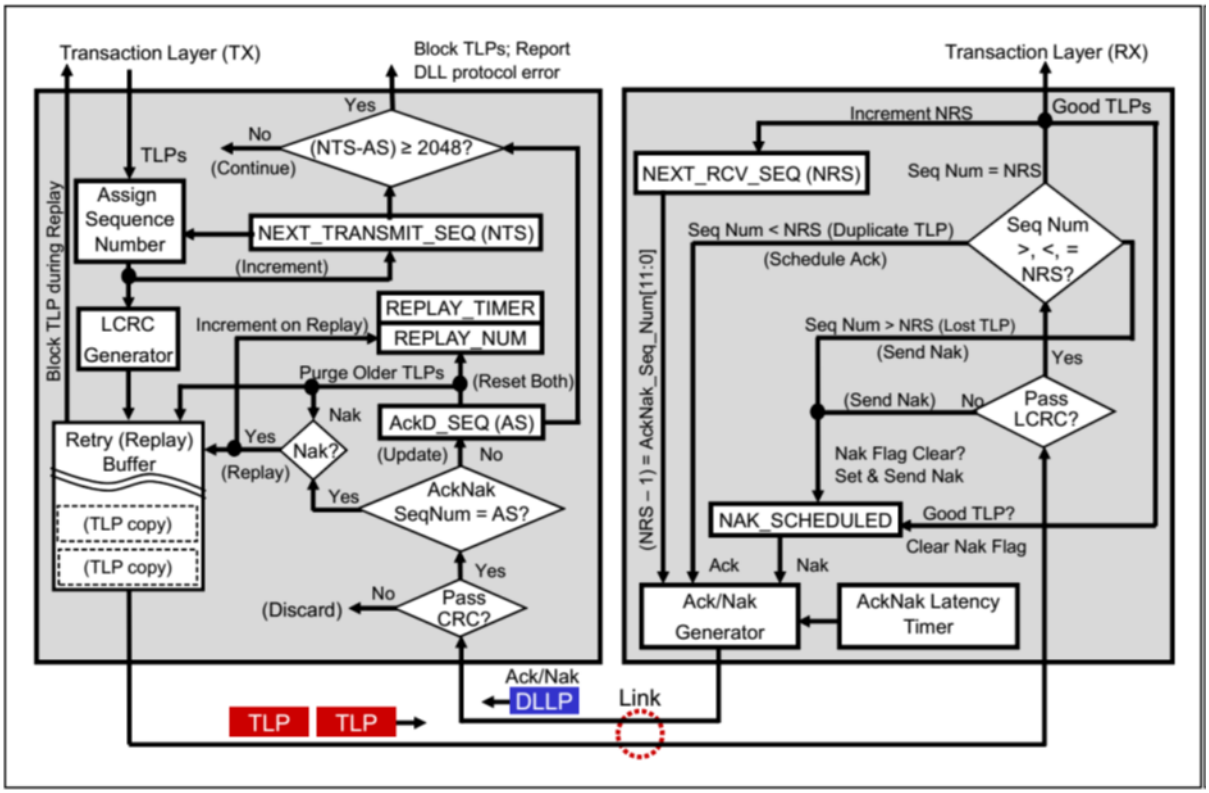
Address routing is based on the target address of TLPs

· Implicit Routing

Message routing is determined using the r[2:0] sub-field of the Type field

TLP Type	Routing Method Used
Memory Read [Lock], Memory Write, AtomicOp	Address Routing
IO Read and Write	Address Routing
Configuration Read and Write	ID Routing
Message, Message With Data	Address Routing, ID Routing, or Implicit routing
Completion, Completion With Data	ID Routing

2.2 Data link layer



2.3 Physical Layer

Physical packet

- Link training and initialization
- Logic sub-layer

Byte striping/un-striping

Scrambling/Descrambling

Encoding/Decoding

Serializing/Deserializing

Ordered sets

TS1 and TS2 Ordered Set (TS1OS/TS2OS)

Link initialization and training

Electrical Idle Ordered Set (EIOS)

Transmitter: lower-power link state sends this before ceasing transmission.

FTS Ordered Set (FTSOS)

Transmitter: Send N_FTS to take Link back to L0 from L0s

SKP Ordered Set (SOS)

Transmitted at regular intervals: Clock Tolerance Compensation

Electrical Idle Exit Ordered Set (EIEOS)

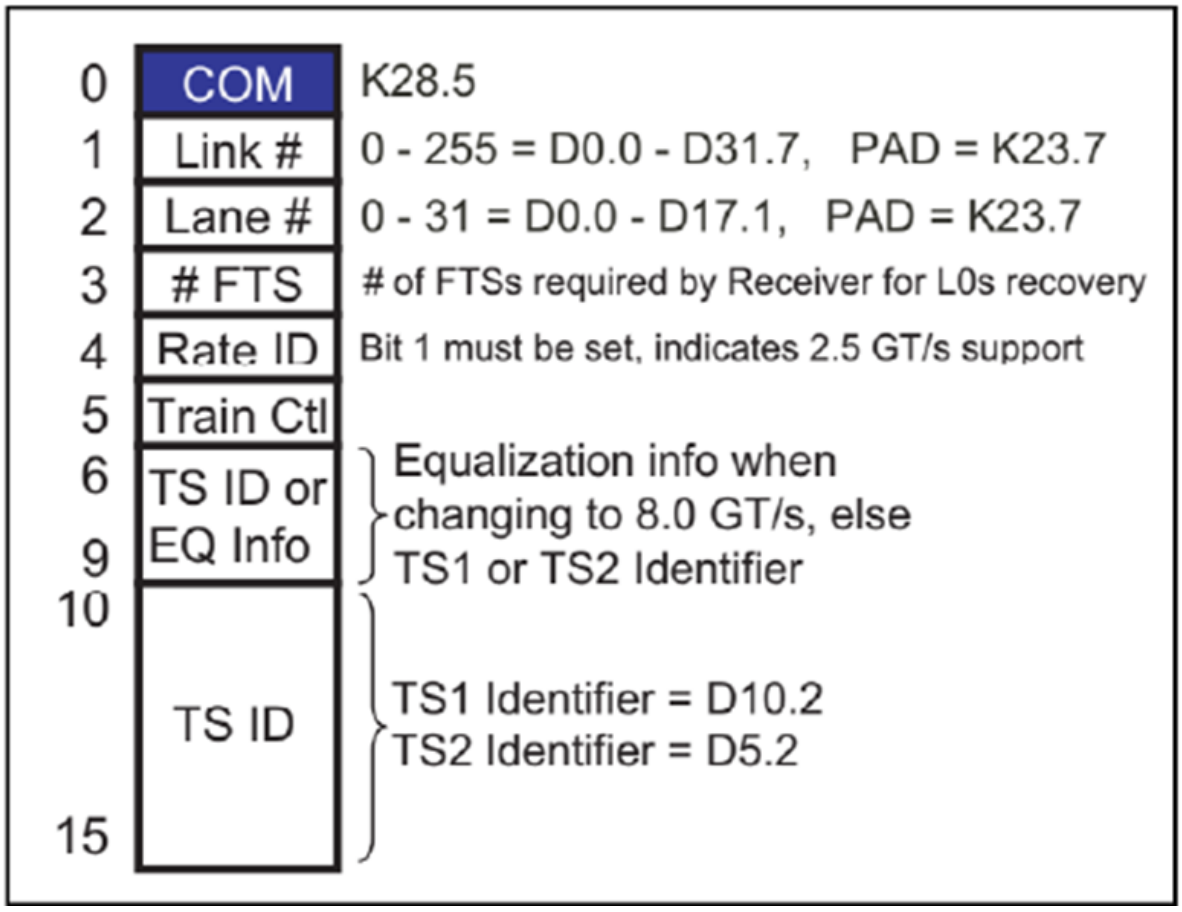
Exit the electrical idle Link state.

Start of Data Stream Ordered Set (SDSOS)

Indicate the following blocks is data stream.

Added in Gen3.

Figure 14-4: TS1 and TS2 Ordered Sets When In Gen1 or Gen2 Mode



Link training and initialization

• Detect − Presence of a receiver

• Polling − Bit Lock

− Symbol Lock/ Block Alignment

− Polarity Inversion

• Configuration − Determine Link width

− Assign Lane numbers

− Lane reversal

− Lane-to-Lane De-skew

• L0 − Fully-active state

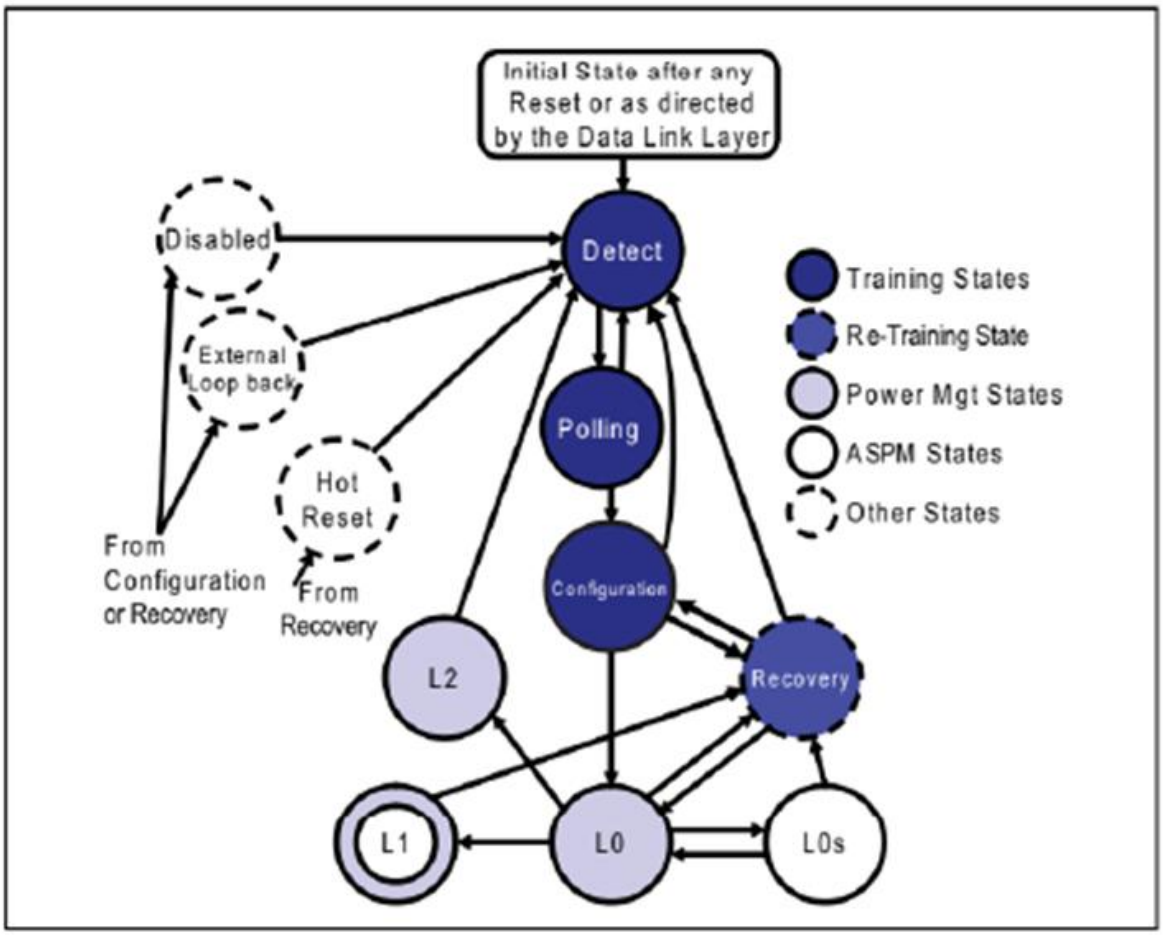
• Recovery − Re-training: L0 error, L1/L0s back to L0, speed change

− Bit Lock

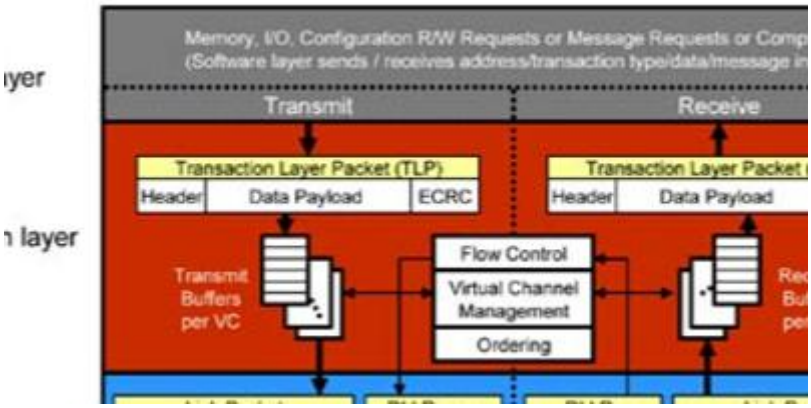
− Symbol Lock/ Block Alignment

− Equalization if support Gen3 or later Generation

Figure 14-6: Link Training and Status State Machine (LTSSM)



Transaction Layer in PCIe Part -1: Overview and TLP



The **Transaction Layer (TL)** in PCIe (Peripheral Component Interconnect Express) is the topmost layer responsible for managing transaction-level communication between devices. It handles packetization, flow control, error detection, and routing.

Overview of the Transaction Layer (TL)

The TL sits between the device's core logic (application layer) and the Data Link Layer (DLL). Its primary roles include:

1. **Packetization**: Assembling and disassembling **Transaction Layer Packets (TLPs)**.
2. **Data Exchange**: Translating application-layer read/write requests into TLPs for transmission.
3. **Error Detection**: Checking for **End-to-End CRC (ECRC)** errors.
4. **Flow Control**: Managing buffer credits to prevent overflow.
5. **Address Space Support**: Handling **Memory, I/O, Configuration, and Message** transactions.

Transaction Layer Packet (TLP) Structure

A TLP consists of:

1. **Header** (3 or 4 Double Words (DW)): **Format Field**: Determines header size and payload presence: **Bit 0**: 0 = 3DW header, 1 = 4DW header. **Bit 1**: 0 = No payload, 1 = Payload included. **Type Field**: Specifies TLP type (e.g., Memory Read, Configuration Write). **Address/Request Details**: Varies by TLP type (e.g., 32-bit vs. 64-bit memory addresses).
2. **Data Payload** (0–1024 DW, configurable).
3. **TLP Digest (Optional)**: ECRC for error detection.

Figure 1 :- TLP packet structure

Byte 0		Byte 1						Byte 2				Byte 3
Format	Type	R	TC	R	Attr	R	TH	TD	EP	Attr	AT	length
Byte 4		Byte 5						Byte 6				Byte 7
(Byte 4-6 depends on type of TLP)												Last DW BE / First DW BE
Byte 8		Byte 9						Byte 10				Byte 11
(Byte 8-11 depends on type of TLP)												
Byte 12		Byte 13						Byte 14				Byte 15
(Byte 12-15 depends on type of TLP(not necessary always))												

Figure 2 :- General TLP header structure

Types of TLPs

TLPs are categorized into three types based on transaction requirements:

1. Posted TLPs

- **No response required** (e.g., Memory Writes, Messages).
- **Examples**: **Memory Write TLP**: Transfers data to a memory address. **Message TLP**: Used for interrupts, power management, or vendor-specific commands.

2. Non-Posted TLPs

- **Require a Completion TLP** (response).
- **Examples**: **Memory Read**: Requests data from a memory address. **I/O Read/Write**: Accesses legacy I/O space (limited to 32-bit, often 16-bit). **Configuration Read/Write**: Accesses PCIe configuration space (registers).

3. Completion TLPs

- **Response to Non-Posted TLPs** (e.g., returning read data).
- **Fields**: **Completer ID**: Identifies the responding device. **Status**: Indicates success/failure (e.g., "Unsupported Request"). **Data Payload**: For read completions.

TLP Header Formats

Each TLP type has a unique header structure:

Memory Request TLP

- **3DW Header**: 32-bit memory address.
- **4DW Header**: 64-bit memory address.
- **Key Fields**: Address: Target memory location. Length: Data payload size (in DW). First/Last DW Byte Enable: Controls byte granularity.

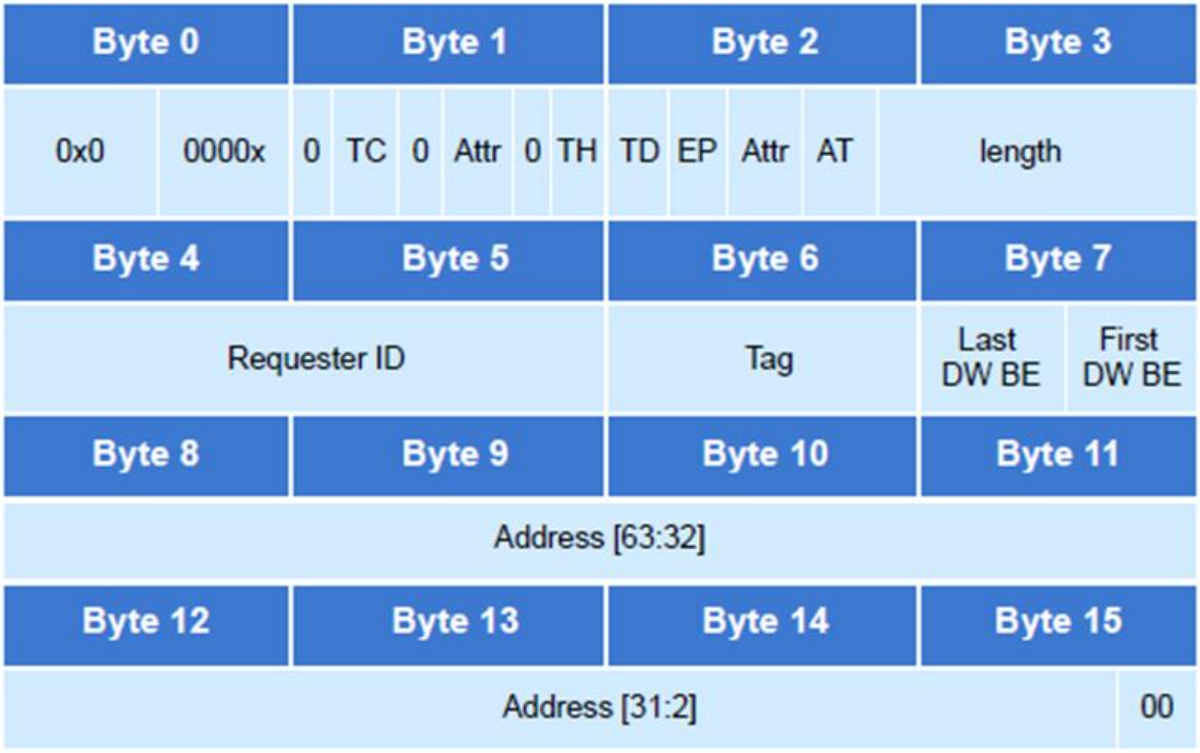


Figure 4 :- 4DW Memory request header

Configuration Request TLP

- Accesses PCIe configuration space (device registers).
- **Key Fields**: Bus/Device/Function (BDF): Identifies the target device. Register Number: Specifies the register offset.

Byte 0		Byte 1	Byte 2			Byte 3
0x0	0010x	00000000	TD	EP	0000	000000001
Byte 4		**Byte 5**	**Byte 6**			**Byte 7**
Requester ID			Tag		0000	First DW BE
Byte 8		**Byte 9**	**Byte 10**			**Byte 11**
Bus number		Device number / Function number	0000	Extended Register number	Register number	00

Figure 5 :- Configuration request header

I/O Request TLP

- Always uses a **4DW header** but supports only **1DW payload** (32-bit data).
- Limited to legacy systems (rarely used in modern PCIe).

Byte 0		Byte 1	Byte 2			Byte 3
0x0	00010	00000000	TD	EP	0000	000000001
Byte 4		**Byte 5**	**Byte 6**			**Byte 7**
Requester ID			Tag		0000	First DW BE
Byte 8		**Byte 9**	**Byte 10**			**Byte 11**
Address [31:2]						00

Figure 6 :- IO request header

Message Request TLP

- **4DW header** with a Message Code field (e.g., interrupt, error signaling).
- Bypasses traditional address spaces.

Byte 0		Byte 1					Byte 2			Byte 3
0x1	10xxx	0	TC	0	Attr	00	TD	EP	0000	length
Byte 4		**Byte 5**					**Byte 6**			**Byte 7**
Requester ID							Tag			Message code
Byte 8		**Byte 9**					**Byte 10**			**Byte 11**
Address [63:32]										
Byte 12		**Byte 13**					**Byte 14**			**Byte 15**
Address [31:2]										00

Figure 7 :- Message request header

Completion TLP

- Matches the original request using Requester ID and Tag.
- Includes Completion Status and data (for reads).

Byte 0		Byte 1					Byte 2			Byte 3
0x0	01010	0	TC	0	Attr	00	TD	EP	0000	000000001
Byte 4		**Byte 5**					**Byte 6**			**Byte 7**
Completer ID							Completion status	BCM		Byte Count
Byte 8		**Byte 9**					**Byte 10**			**Byte 11**
Requester ID							Tag		0	Lower address

Figure 8 :- Completion header

TLP Flow in PCIe

Transmitter:

- TL forms TLP (header, payload, ECRC if enabled).
- Passes to DLL, which adds sequence number and LCRC.
- PL adds STP (start), END, and serializes data.

Receiver:

- PL deserializes, removes STP/END.
- DLL verifies LCRC, sequence number, and sends TLP to TL.
- TL checks ECRC, extracts data, and delivers to core logic.

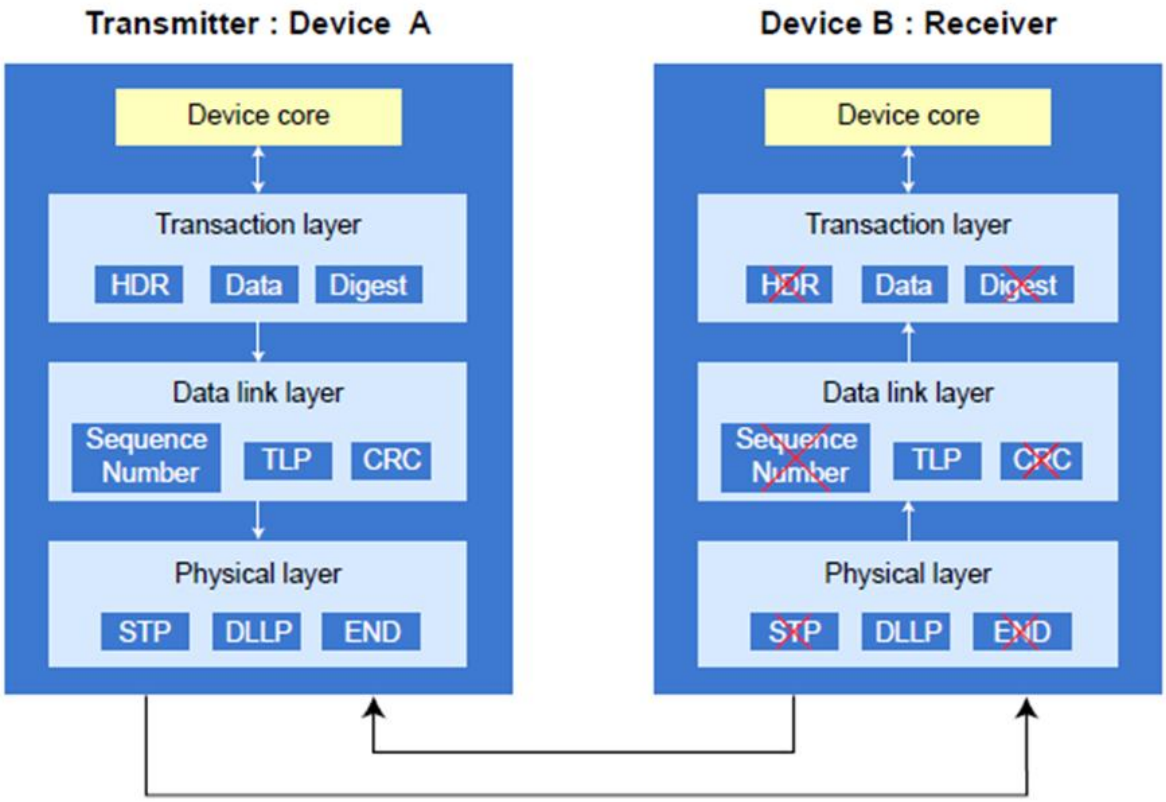


Figure 9 :- Data transmission in PCIe

Flow Control in the Transaction Layer

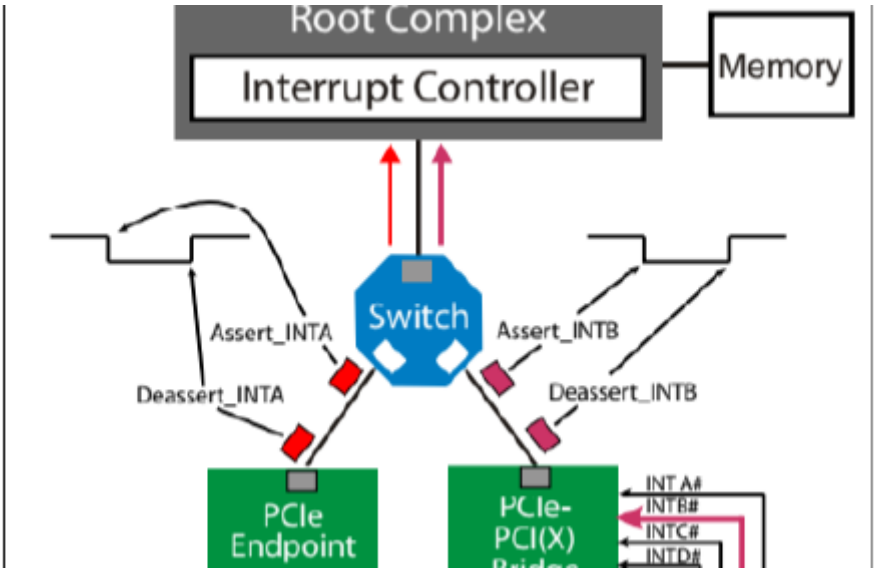
- **Credit-Based Mechanism**: The receiver reports buffer availability (**credits**) via **Data Link Layer Packets (DLLPs)**. Virtual Channels (VCs): Up to 8 independent channels for prioritizing traffic (e.g., high-priority vs. bulk data).
- **Process**:
- Transmitter checks receiver credits.

- Credits are updated via DLLPs exchanged between DLL layers.
- TLPs are sent only if credits are available.

Generic Header Fields:

- **Format** (Fmt): header size is 3 DW or 4 DW, payload presence.
- **Type**: Encodes the transaction type.
- **TC** (Traffic Class): Priority level (0–7) for QoS.
- **Attr** (Attributes): Controls ordering (Relaxed, ID-Based) and cache behavior (No Snoop).
- **TH** (TLP Processing Hints): 1 bit.
- **TD** (TLP Digest): 1 bit (ECRC presence).
- **EP** (Poisoned Data): 1 bit: Marks corrupted data (e.g., for error handling).
- **Length**: 10 bits (payload size in DW).
- **Requester ID**: Bus/Device/Function (BDF) of the transaction initiator.
- **Tag**: Unique ID for tracking non-posted transactions.
- **Completion ID**: Bus/Device/Function (BDF) of the transaction completer.
- **Transaction ID**: The combination of the Requester ID and the Tag field of the TLP is termed as Transaction ID of the TLP.

PCIe enumeration and Interrupts Mechanism:



PCIe enumeration and resource assignment

PCI/PCIe has 3 address spaces:

Configuration Space

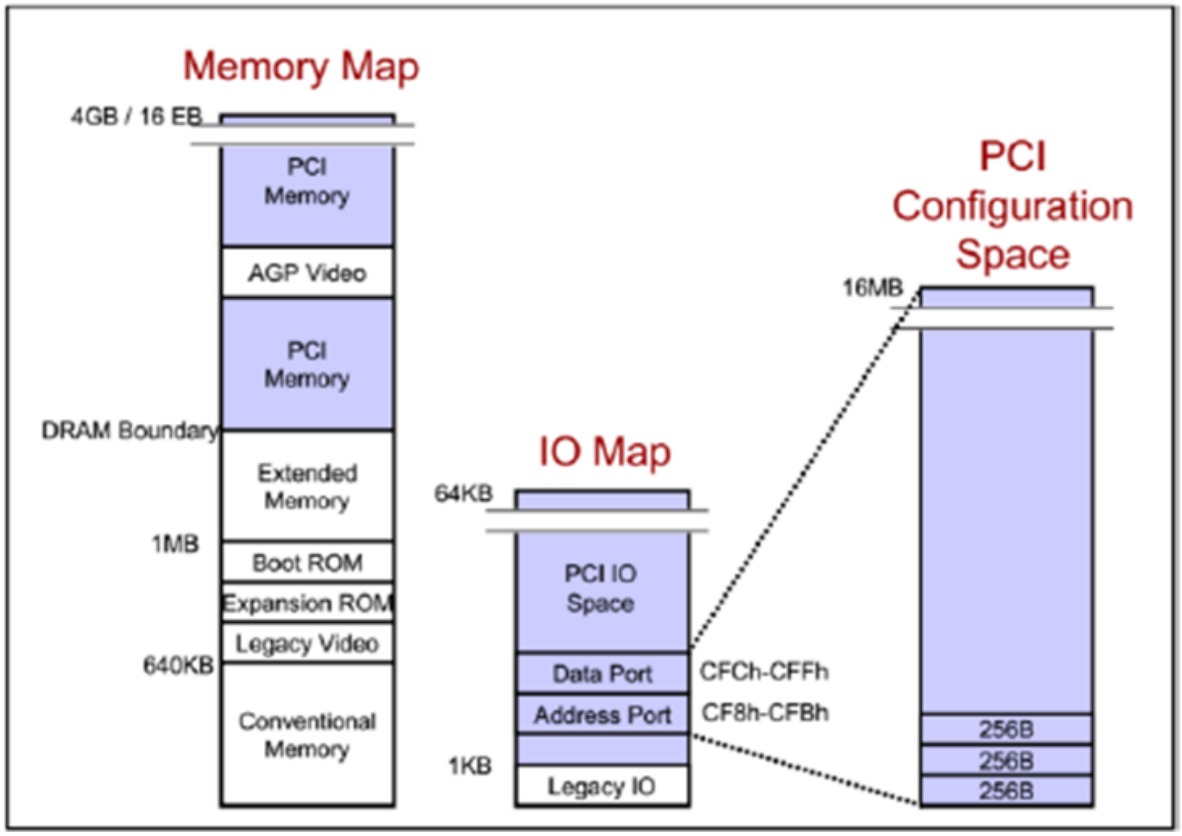
PCI function: 256B

PCIe function: 4KB

Memory Space (32bit/64bit)

IO Space (16bit/32bit)

x86 has 16bit IO space

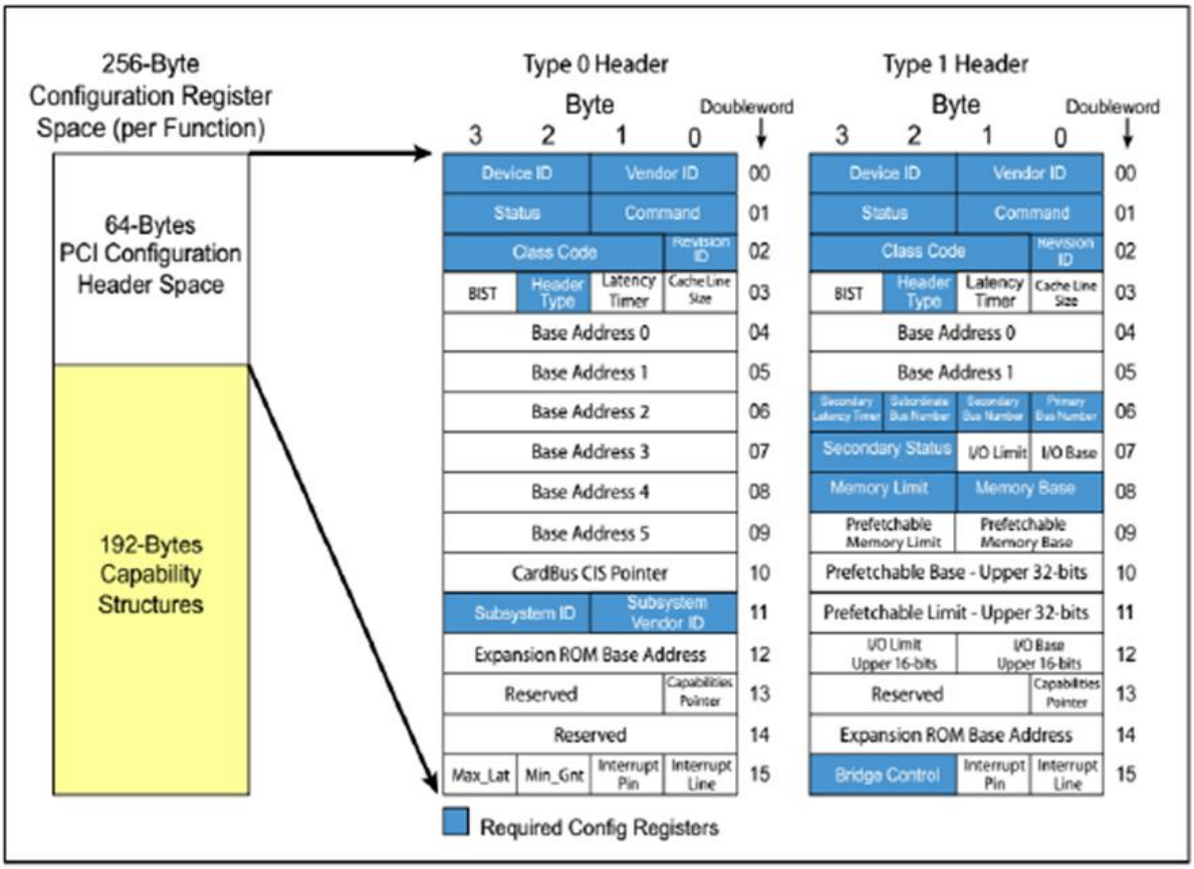


PCI/PCIe Configuration register space

PCI Compatible Configuration Space: 256B

PCIe Configuration Space extend to 4KB

Figure 3-2: PCI Compatible Configuration Register Space



PCIe enumeration process and resource assignment

• Enumeration start from BDF 0:0.0, read the Vendor&Device ID to check if the located device exists.

• If exist, allocate a pci_dev instance and do further setup such as calculate the size of BARs (according to the device's header type).

• If it's a multiple function device, scan and do the same setup for all the implemented functions.

• If the device is a PCI bridge, setup the Primary/Secondary bus number and enumerate the child bus, update Subordinate bus number.

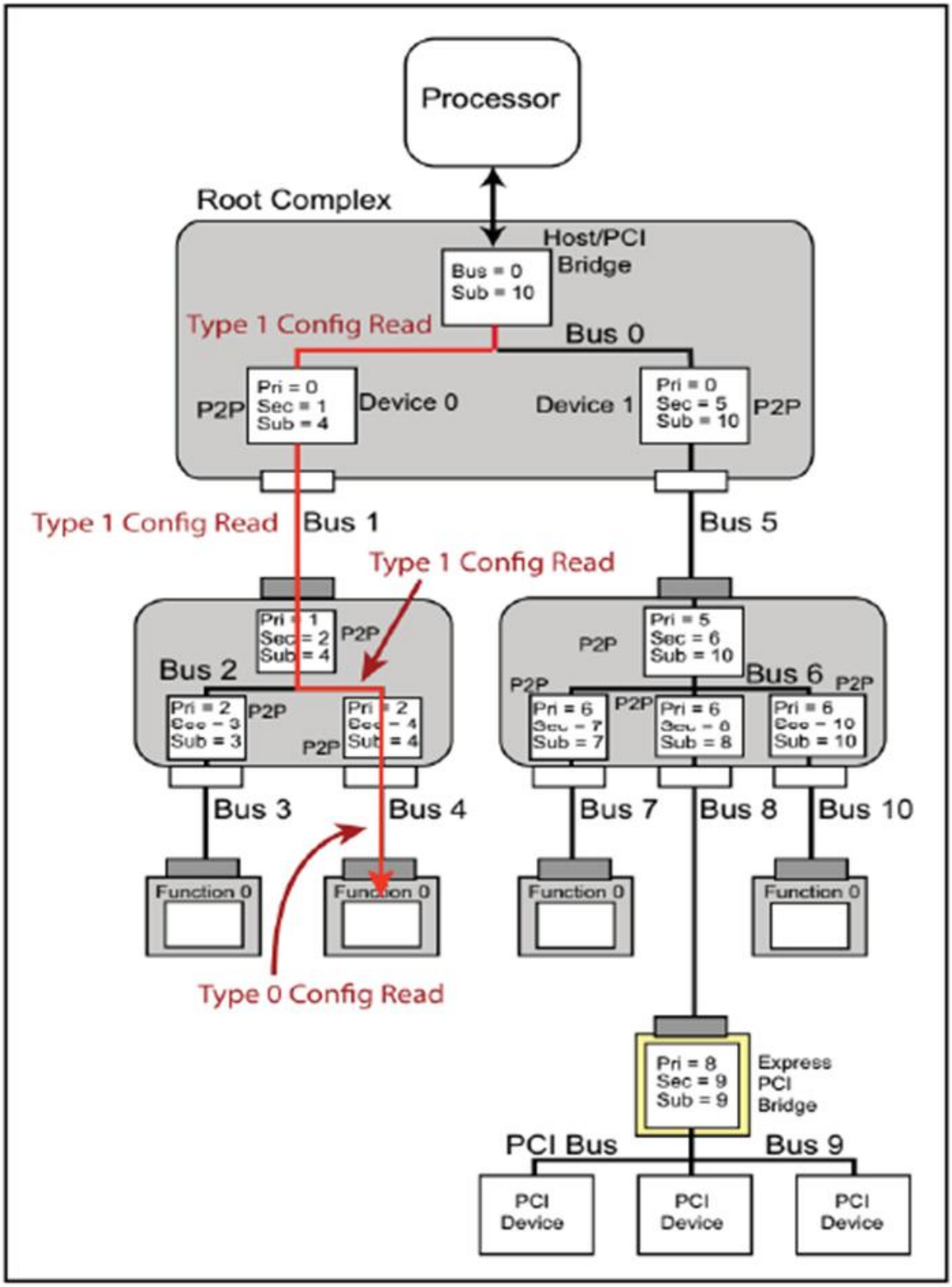
• Enumeration stop when scanned all the BDFs.

• Assign resource to the PCI devices from root bus.

• First update the BARs of the devices directly connected.

• If PCI bridge, iterate each subordinate PCI device to calculate the size of IO, MEM, PRE-MEM resources and update the IO Base/Limit, Memory Base/Limit and Prefetchable Memory Base/Limit Registers.

- Then update the BARs of devices connected to the child bus.
- Depth first recursion.



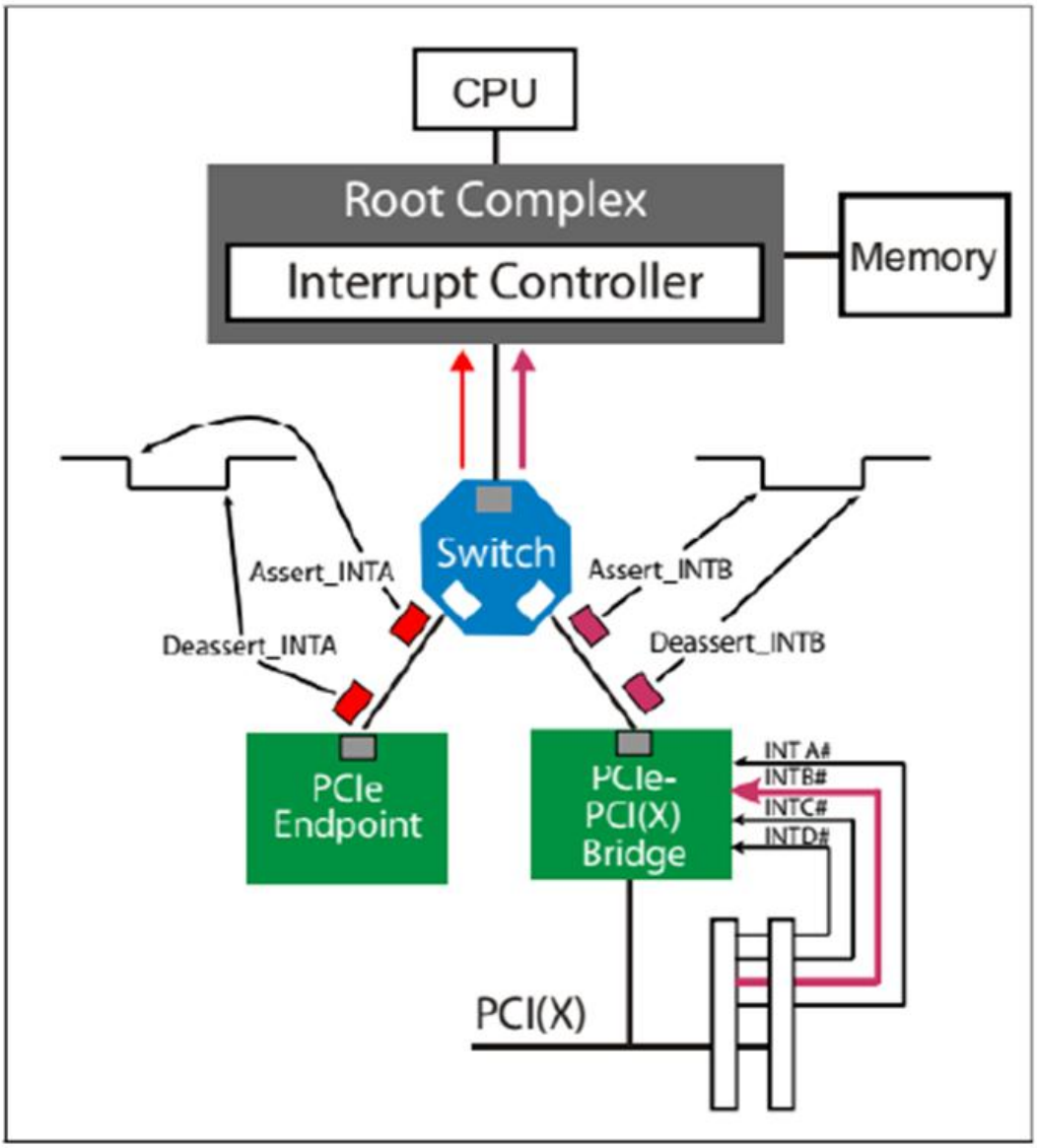
Interrupts Mechanism

INTx (Legacy interrupt)

PCI/PCI-X: side-band signal

PCIe: in-band signal (INTx Message)

Figure 17-9: Example of INTx Messages to Virtualize INTA#-INTD# Signal Transitions

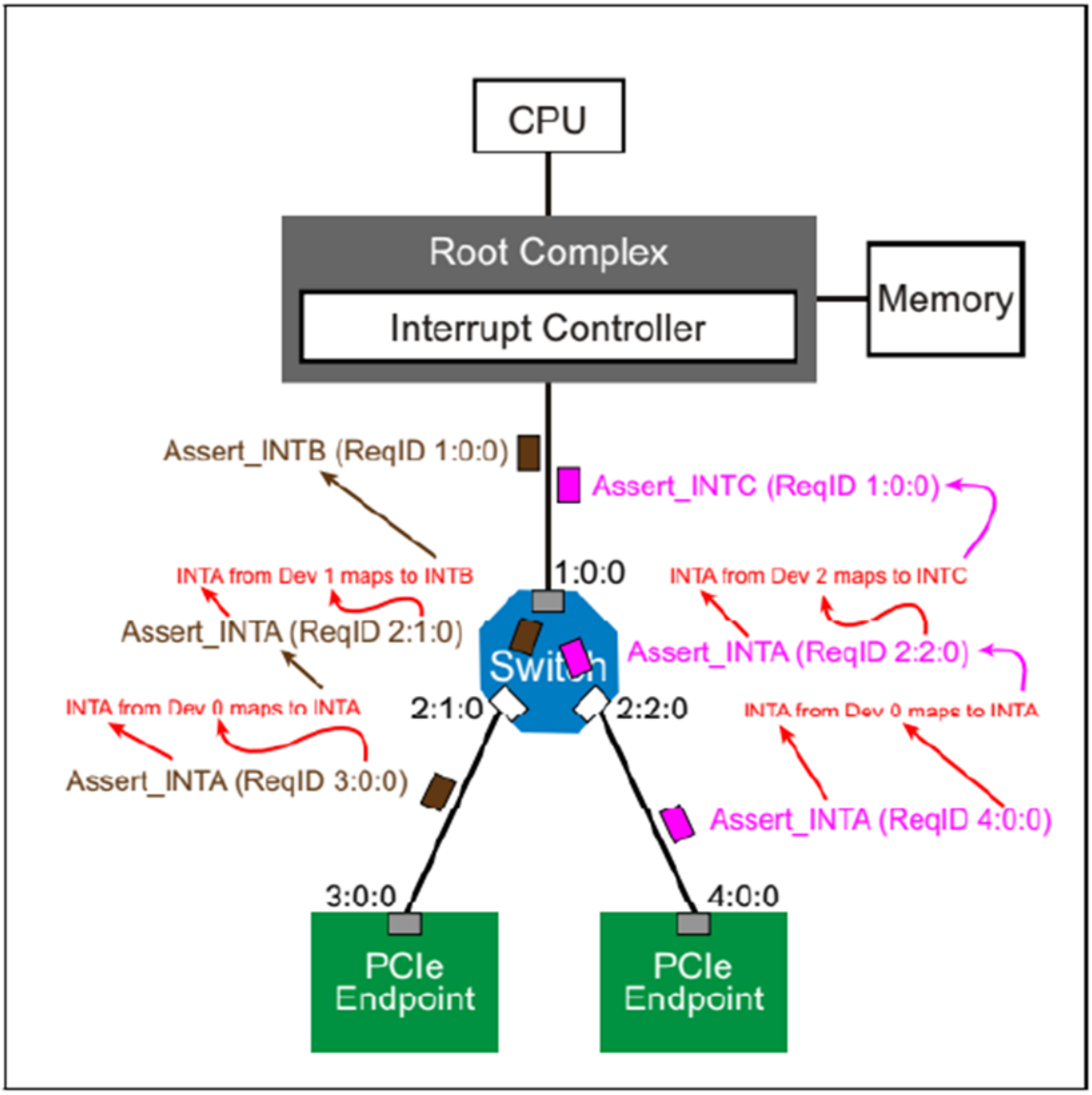


PCIe switch: INTx swizzle

• Interrupt balance

• ISR Latency

Figure 17 11: Example of INTx Mapping



MSI - Message Signalled Interrupt

• Transaction Type

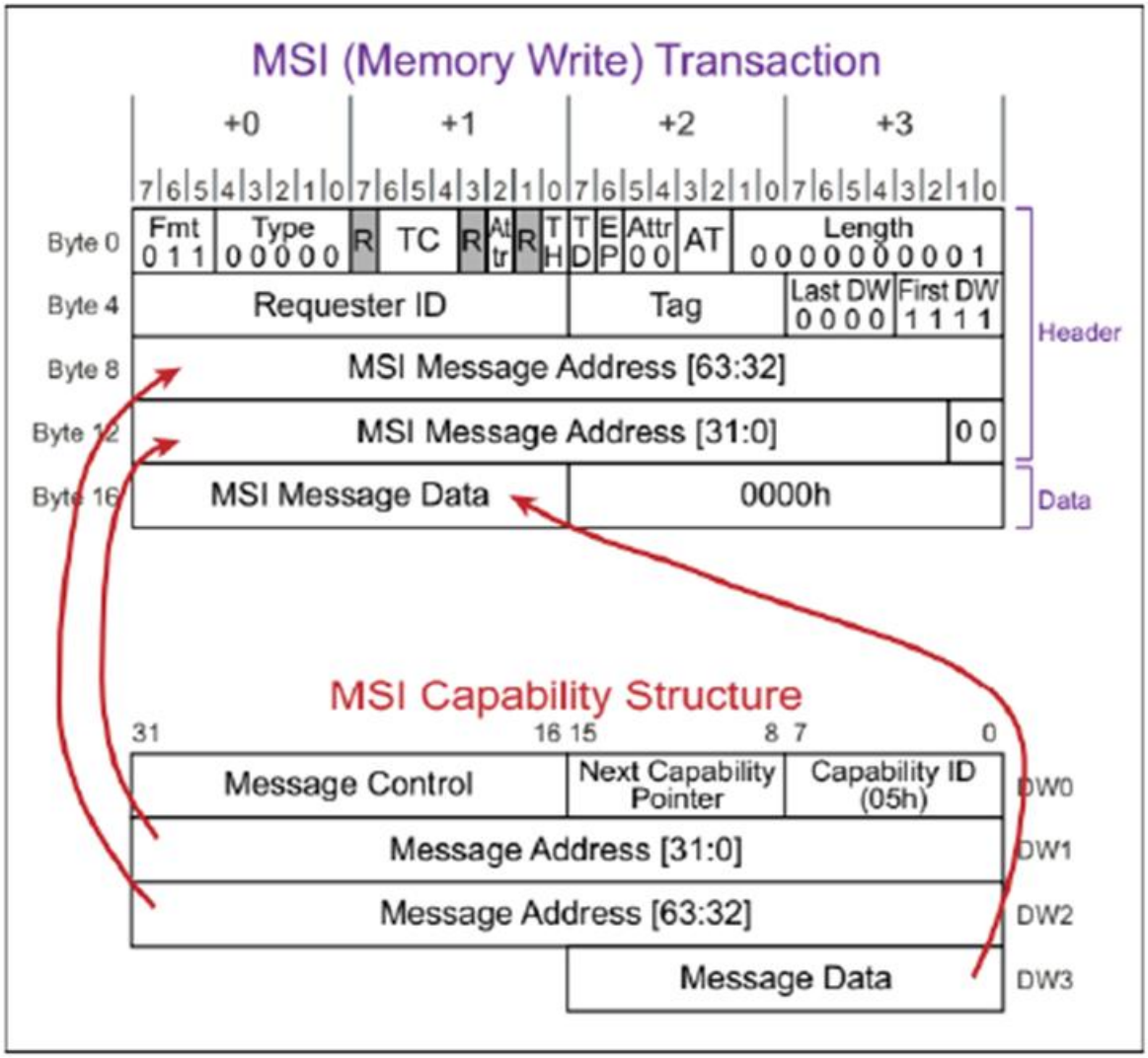
• Memory Write TLP

• Interrupt number: Multiple MSI up to 32

• Single Message Address

• Message Data is changed by device.

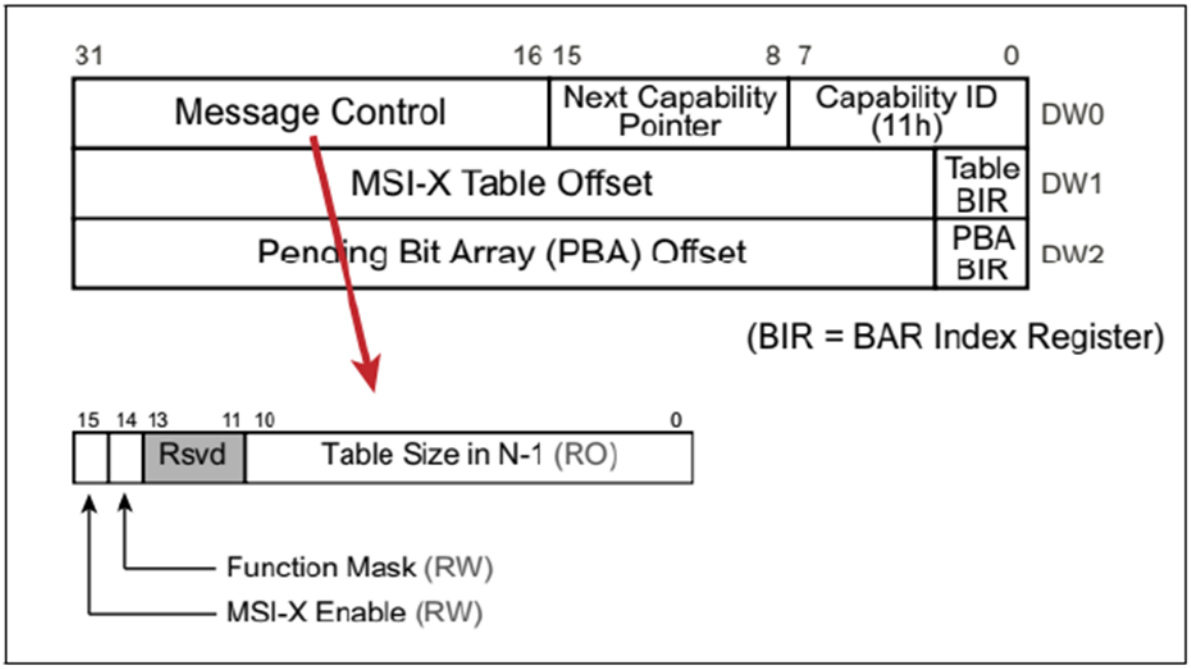
Figure 17-16: Format of Memory Write Transaction for Native-Device MSI Delivery



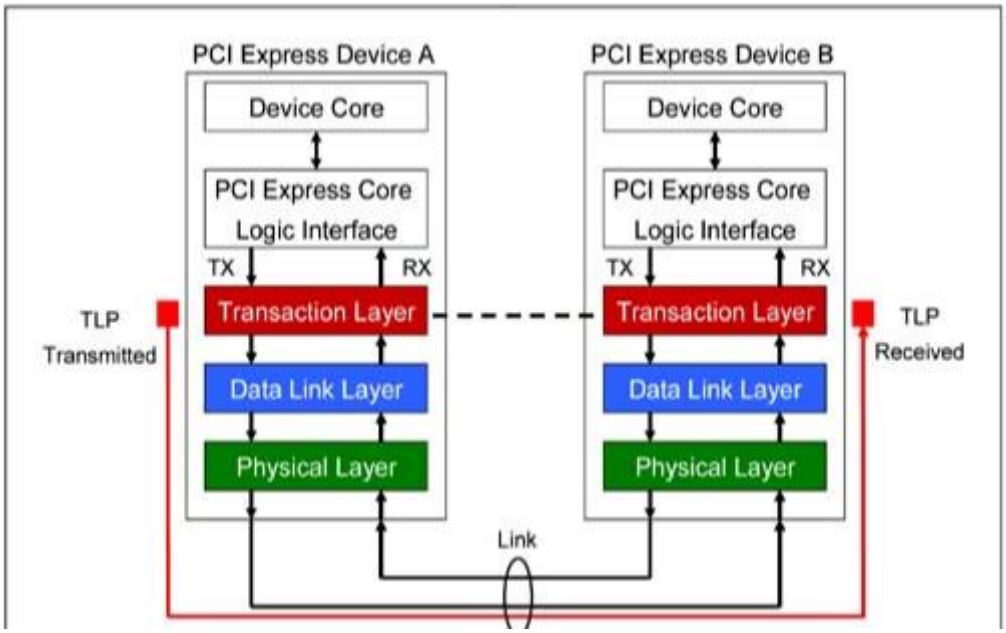
MSI-X

- Transaction Type
  - The same as MSI
- Interrupt number
  - Up to 2048
- MSI-X table and PBA (Pending Bit Array) reside in BAR

Figure 17-17: MSI-X Capability Structure



Features added in PCIe Gen 3:



PCIe Gen 3 Standard Development

Double the effective bandwidth of PCIe Gen 2 (from 5 GT/s) while maintaining or reducing power consumption and ensuring backward compatibility.

1. **Increased Data Rate & Encoding Efficiency: Signalling Rate:** Increased from 5 GT/s (Gen 2) to **8 GT/s** (Gen 3). **Encoding Scheme:** Replaced **8b/10b** (20% overhead) with **128b/130b** (1–2% overhead), boosting effective bandwidth to **~7.99 GT/s** (nearly double Gen 2).
2. **Technical Challenges Addressed: Tighter Timing Requirements:** Clock timing reduced from 200ps to 125ps, jitter tolerance tightened from 44ps to 14ps, and SSC shared bandwidth lowered from 80ps to 35ps. **Signal Integrity:** System, board, and package modelling ensured feasibility for mobile (8"), client (14"), and server (20") channel reaches.
3. **Power Efficiency:** Delivers **twice the data rate** of Gen 2 with **equal or lower power consumption**.
4. **Backward Compatibility & Flexibility:** Retains **100MHz reference clock** and existing channel lengths for seamless migration. Devices **dynamically negotiate** data rates (Gen 1/2/3) based on signal conditions.
5. **Enhanced Signalling Features: Optional Advanced Equalization:** Decision feedback equalization for extended backplane reach. **Improved Components:** Transmitter de-emphasis, receiver equalization, and optimized Tx/Rx Phase Lock Loops (PLLs) and Clock Data Recovery (CDR).

TLP Processing Hints (TPH)

TLP Processing Hints (TPH) improve I/O performance in complex memory hierarchies by optimizing c **Mechanism**:

- **Snoop Filters**: Used in processor chipsets to reduce memory latency and increase throughput by maintaining cache coherence with PCIe memory requests.
- **TPH Structure**: Three bits in the TLP header indicate the presence of TPH and provide processing hints. An optional 8-bit "steering tag" allows system-specific information for better cache management.

Benefits:

- Enables optimal allocation of the cache hierarchy.
- Reduces memory access latency, interconnect overhead, and power consumption.

ID-Based Ordering (IDO)

ID-Based Ordering (IDO) enhances system performance by allowing independent processing of data streams based on Requester ID, avoiding bottlenecks caused by strict or relaxed PCIe ordering rules.

Mechanism:

- **Requester ID Separation**: Multiple data flows are separated by Requester ID, enabling independent processing.
- **Combination with Relaxed Ordering (RO)**: IDO works alongside RO to prevent blocking and improve throughput in multi-function devices and switches.
- **Default State**: IDO is disabled by default but can be enabled by drivers or software if supported.

Benefits:

- Reduces performance bottlenecks in systems with multiple data streams.
- Improves delivery speed of TLP streams from different devices or functions within a device.

Atomic Operations (AO)

Atomic Operations (AO) enhance synchronization in emerging applications like math co-processing, visualization, and content processing, enabling higher performance without relying on interrupt mechanisms.

Mechanism:

- **TLP Header Enhancement**: Adds a few bits to the TLP header to support atomic operations.
- **Defined Operations**: **FetchAdd**: Atomically fetches and adds a value. **Swap & Compare**: Atomically swaps values based on a comparison. **Swap**: Atomically swaps values.

Benefits:

- Improves synchronization efficiency in high-performance applications.
- Reduces the need for interrupt-based synchronization, lowering overhead.

Multicast (MC)

PCIe Multicast enhances efficiency in expanding applications (communications, embedded systems, storage, multi-graphics, imaging) by enabling a single packet to reach multiple destinations simultaneously.

1. **Address-Based Routing:** Utilizes Multicast Base Address Registers (MC BARs), capability structures, and overlay mechanisms.
2. **Transaction Support:** Limited to posted address-routed transactions (e.g., memory writes) for both root complexes (initiators) and endpoints (targets).
3. **Scalability:** Supports up to **64 multicast groups** (as per the Engineering Change Notice, ECN).

Dynamic Power Allocation (DPA)

Enhances power management for high-speed, complex PCIe devices by enabling granular control over power consumption while maintaining performance.

1. **Granular Power States:** Defines **32 sub-states** within the active device power state (D0), allowing dynamic adjustment of power usage without transitioning to lower power.
2. **Latency Management:** Includes specifications to minimize delays when switching between power states, ensuring efficient transitions without significant performance impact.
3. **Active-State Optimization:** Focuses on power management *during active operation* (D0), complementing existing PCIe Gen 2 PM features like dynamic link-speed/width adjustments.

Summary PHY of PCIe Gen 3 Features:

. Data Handling & Encoding

- **Block Synchronization (Word Aligner):** Aligns serial data to 130-bit word boundaries using ordered sets (e.g., EIEOS, SKP). Adjusts dynamically for variable-length SKP ordered sets.
- **128B/130B Encoder/Decoder:** Replaces Gen 1/2's 8B/10B encoding, reducing overhead to 1–2%. Uses a 2-bit sync header + 128-bit data, with scrambling for data packets (excludes ordered sets).
- **Gear Box:** Converts 130-bit blocks (or variable-length SKP sets) into 32-bit segments for PMA serialization, resolving fractional bit mismatches.
- **Scrambler/Descrambler:** Ensures clock transitions via scrambling (excludes sync headers, ordered sets, and TS1/TS2 symbols). Resets seed on Electrical IDLE exit.

Clock & Speed Management

- **Rate Match FIFO:** Compensates for ±300 ppm clock differences by inserting/deleting SKP characters to prevent FIFO overflow/underflow.
- **Auto-Speed Negotiation (ASN):** Dynamically switches between Gen1 (2.5 GT/s), Gen2 (5 GT/s), and Gen3 (8 GT/s) via glitch-free PMA/PCS clock adjustments. Master channel coordinates speed changes in bonded links. Process: PCS reset → PMA rate change → confirmation → PCS reactivation.
- **CDR Control Block:** Manages clock recovery alignment and deskew, ensuring fast lock times (≤4 ms for Gen3 L0s exit).

Power Management

- **Electrical Idle Handling: Transmitter:** Sends EIOS (e.g., 0x66 symbols for Gen3) before entering low-power states. **Receiver:** Detects idle via link inactivity or ASN processes (per PCIe Base Spec 3.0).
- **Power State Management:** Minimal power-saving in Gen3 PCS (e.g., transmitter electrical idle in low states). PIPE block clocks remain active in P2 state.

Interface & Control

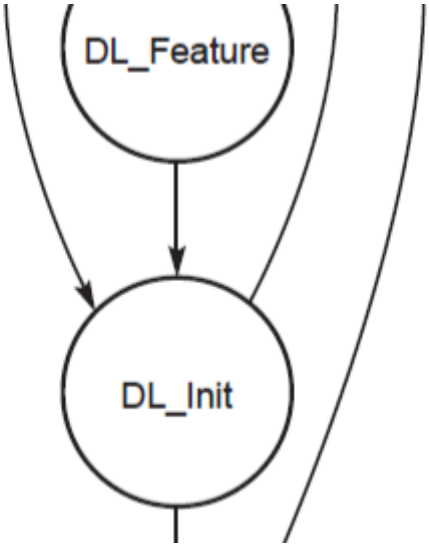
- **PIPE 3.0-Like Interface:** 32-bit wide interface for PHY control (e.g., speed negotiation, electrical idle). Supports dynamic clock switching (62.5–250 MHz) and auto-speed negotiation. Manages 128B/130B encoding, CDR, and transceiver PMA/PLL configurations.

Key Improvements Over Gen 2:

- **Higher Bandwidth:** 8 GT/s signaling with 128B/130B encoding doubles effective bandwidth (7.99 GT/s vs. Gen2's 4 GT/s).
- **Backward Compatibility:** Retains 100 MHz clock and channel lengths (8–20 inches).

- **Power Efficiency:** Double data rate at equal/lower power.
- **Signal Integrity:** Advanced equalization (e.g., decision feedback) and tighter jitter tolerance (14ps vs. 44ps).

Features added in PCIe gen4:



Higher Speed of 16GT/s.

Extended Capability for new lane margining feature

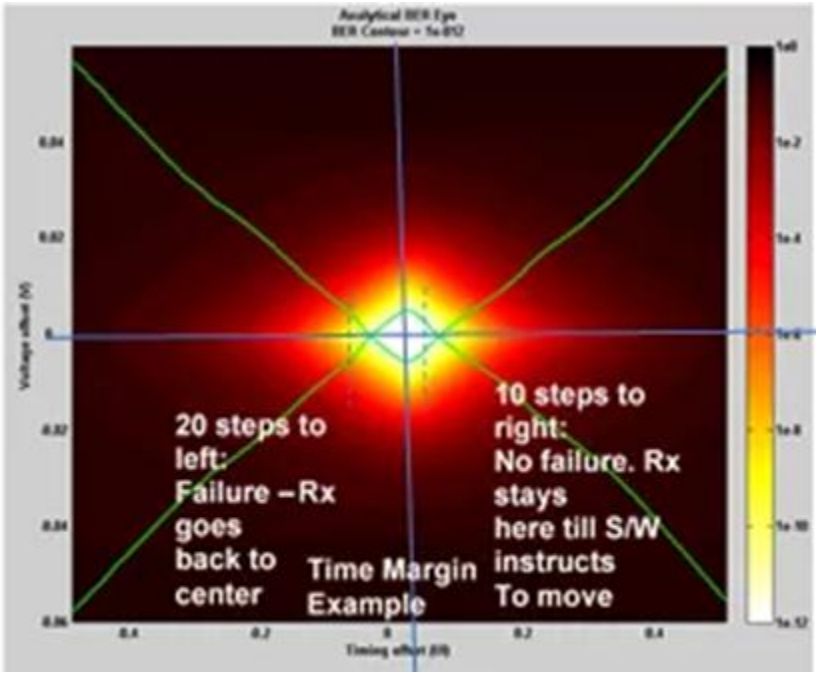
PCIe 4.0 doubled the data rate to **16 GT/s per lane**, introducing challenges for signal integrity due to increased channel loss and noise. To ensure reliable operation, the **Lane Margining at the Receiver** feature was standardized. This allows system designers to measure the "safety margin" of signal integrity in real-world systems, ensuring robustness while maintaining backward compatibility.

Purpose

- **Margin Measurement**: Determines how close a PCIe lane is operating to its functional limits by testing the available "signal eye" (the valid region for data sampling).
- **Standardized Method**: Provides a uniform way to measure margin across all PCIe 4.0+ devices, replacing vendor-specific solutions.

How It Works

- **Sampling Point Adjustment**: The receiver shifts its sampling point horizontally (timing) and/or vertically (voltage) within the signal eye to detect the point where errors begin. Example: An asymmetric eye might show 10 steps to the right (timing margin) and 20 steps to the left before failure.
- **Non-Destructive Testing**: Runs during normal operation (**L0 state**), allowing margin measurement without disrupting live traffic.



Benefits

- **Production-Ready Validation**: Works on real platforms without external test equipment. Enables batch testing to detect material/process variations.
- **Proactive Monitoring**: Track margin degradation over time to predict failures (e.g., due to aging or temperature changes).
- **Retimer Support**: Standardizes control of retimers (devices that extend PCIe reach), ensuring end-to-end margin visibility.
- **Standardized Registers**: Uses architected registers for control/reporting, ensuring interoperability across vendors.

Challenges Addressed by Lane Margining

- **Manufacturing Variability**: High-volume production introduces variations in process, voltage, and temperature (PVT), affecting signal quality.
- **Retimer Complexity**: Retimers amplify and reshape signals but complicate margin analysis; Lane Margining standardizes their control.
- **Real-World Noise**: Electrical noise (e.g., from power supplies or crosstalk) impacts signal integrity in live systems.
- **Proprietary Limitations**: Pre-PCIe 4.0 solutions lacked standardization, making cross-vendor validation difficult.

Robust Equalization:

· Unlike PCIe Gen 3, which used **static equalization**, Gen 4 implements **dynamic equalization**.

· The **receiver (Rx) negotiates with the transmitter (Tx)** to **find the optimal equalization settings**.

· This is done **in real-time**, adjusting **based on channel conditions**.

Key Equalization Techniques Used in PCIe Gen 4:

· **Feed-Forward Equalization (FFE) at Tx** – Applies **pre-emphasis** to compensate for high-frequency losses.

· **Continuous-Time Linear Equalization (CTLE) at Rx** – Filters the signal and amplifies specific frequencies.

· **Decision Feedback Equalization (DFE) at Rx** – Uses past symbol decisions to correct future signals.

Steps in PCIe Gen 4 Link Training:

1. Receiver detects incoming signal and measures distortion.
2. Process repeats until the best signal integrity is achieved.
3. Receiver sends requests to transmitter to adjust equalization settings.
4. Transmitter fine-tunes pre-emphasis and equalization settings.

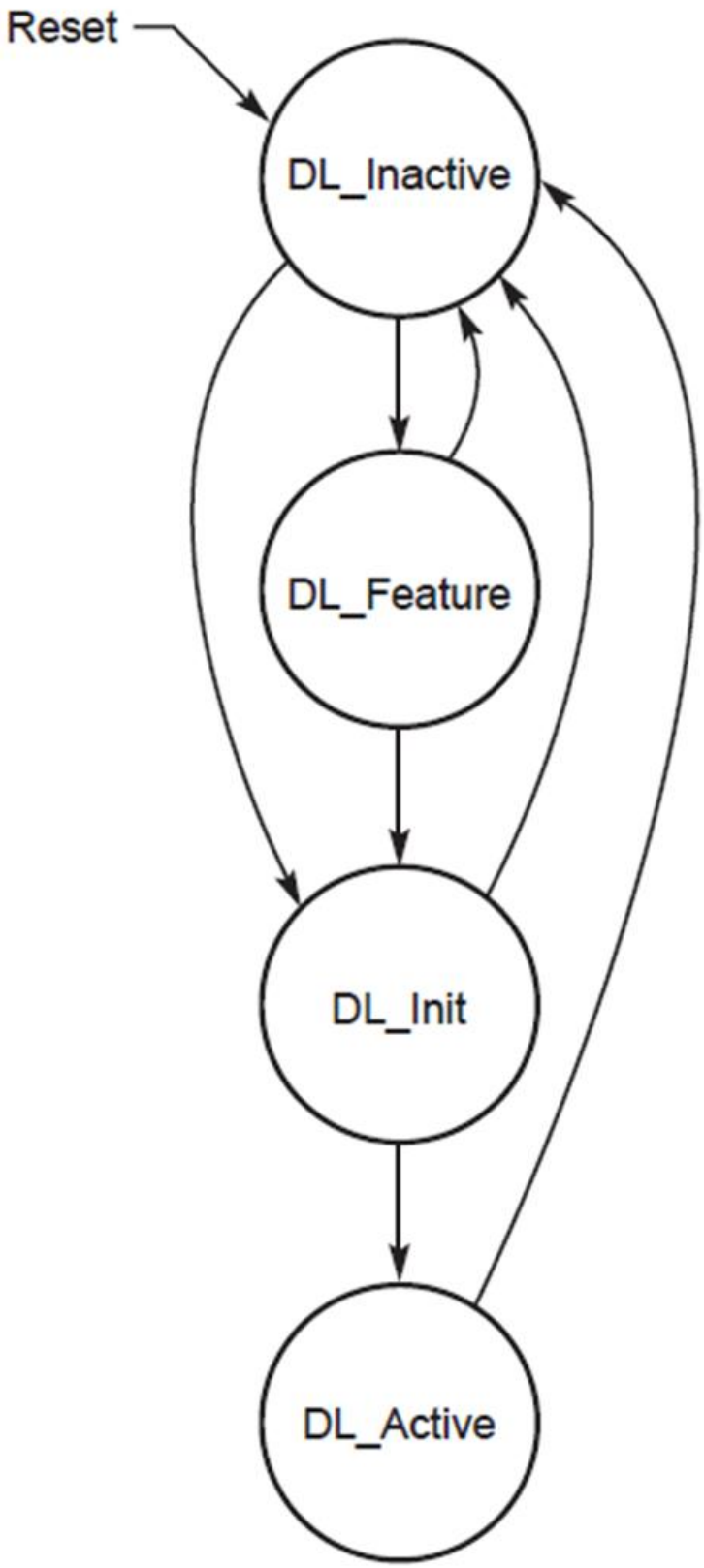
New DLL Feature State Added to DLCMSM in PCIe

The **Data Link Control and Management State Machine (DLCMSM)** is responsible for **managing the link layer operations**, including error handling, data integrity, and feature negotiation.

With newer PCIe specifications a **new Data Link Feature state** has been introduced in DLCMSM. This state activates when a specific feature bit is **set in both the Local and Remote Data Link Feature Supported fields**.

Table 3-1 Data Link Feature Supported Bit Definition

Bit Location	Description
0	Scaled Flow Control - indicates support for Scaled Flow Control. Scaled Flow Control must be supported in Ports that support 16.0 GT/s or higher operation.
22:1	Reserved



e 3-2 Data Link Control and Management State Machine

NOP DLLP Transmission for LCRC Checking in PCIe

The **NOP (No Operation) Data Link Layer Packet (DLLP)** is a special type of DLLP introduced in PCIe for **LCRC (Link CRC) checking** and link testing.

1. **Reserved DLLP Type Encoding**

- NOP DLLP has a **dedicated encoding** in the PCIe specification.
- It is designed to be ignored by receivers during normal operation.

2. **Ensures LCRC Integrity**

- While NOP DLLPs **carry no actual data**, they are still subject to **LCRC (Link CRC) computation**.
- This allows the PCIe link to **verify LCRC correctness without impacting actual data transmission**.

Simplified Replay Timer in PCIe Gen 4

The **simplified replay timer** in PCIe Gen 4 is an improvement over previous generations, designed to enhance **data retransmission efficiency** while reducing system complexity. This mechanism ensures that retransmissions due to **link errors** occur with minimal latency and computational overhead.

10-Bit Tag in PCIe Gen 4 for Increasing Outstanding TLPs

In **PCIe Gen 4**, the **transaction tag field** was expanded from **8 bits to 10 bits**, allowing a significantly larger number of **outstanding Transaction Layer Packets (TLPs)**. This change **improves performance and efficiency** for systems handling high-bandwidth workloads.

Increased Tag Space for More Outstanding Transactions

- **PCIe Gen 3 used an 8-bit tag field**, limiting the number of **simultaneous non-posted transactions (like reads and completions)** to **256**.
- **PCIe Gen 4 increased this to 10 bits**, allowing **more outstanding transactions**.
- This expansion is crucial for devices that require **multiple simultaneous data exchanges** without waiting for previous requests to complete.

Improved Throughput & Parallelism

- A higher number of **simultaneous transactions** improves overall PCIe link efficiency, **reducing idle time**.

- This is **especially beneficial in data center applications, high-performance computing (HPC), and graphics workloads**, where a large number of memory and I/O transactions occur at once.

Scaled Flow Control in PCIe Gen 4

Scaled Flow Control is a **dynamic credit allocation system** introduced in **PCIe Gen 4** to improve **throughput efficiency**. Unlike previous generations that used **fixed flow control credits**, PCIe Gen 4 enables the sender and receiver to **dynamically adjust the number of available credits** based on **current link conditions**. This ensures **optimized bandwidth utilization** and prevents **data congestion**.

MSI-X with Steering Tag Feature in PCIe

The **MSI-X with Steering Tag** feature in **PCIe Gen 4** enhances **Message-Signaled Interrupts (MSI-X)** by introducing a **Steering Tag mechanism**. This improves **interrupt handling flexibility** and **system performance**, particularly in high-throughput environments like data centers, GPUs, and storage systems.

Overview of MSI-X

MSI-X is an extension of **MSI (Message-Signaled Interrupts)** that allows devices to generate **interrupts without dedicated pins** by writing a value to a specific memory address. This improves scalability by supporting a **larger number of interrupt vectors** compared to MSI.

Key Benefits of MSI-X Over MSI

· Supports up to **2048 unique interrupt vectors** (compared to 32 in MSI).

· Allows independent configuration of **interrupt vectors per device function**.

· Enables per-vector **masking**, providing more granular interrupt control.

What is the Steering Tag?

The **Steering Tag** is an additional **identifier** used within MSI-X to enhance interrupt processing efficiency.

How the Steering Tag Works

- It is **associated with the Hint field** in MSI-X.
- When an interrupt is generated using **cfg_interrupt_msix_int**, the **Steering Tag helps direct the interrupt** to a specific CPU core or processing unit.
- This is **useful in multi-core systems** where efficient interrupt routing is critical.

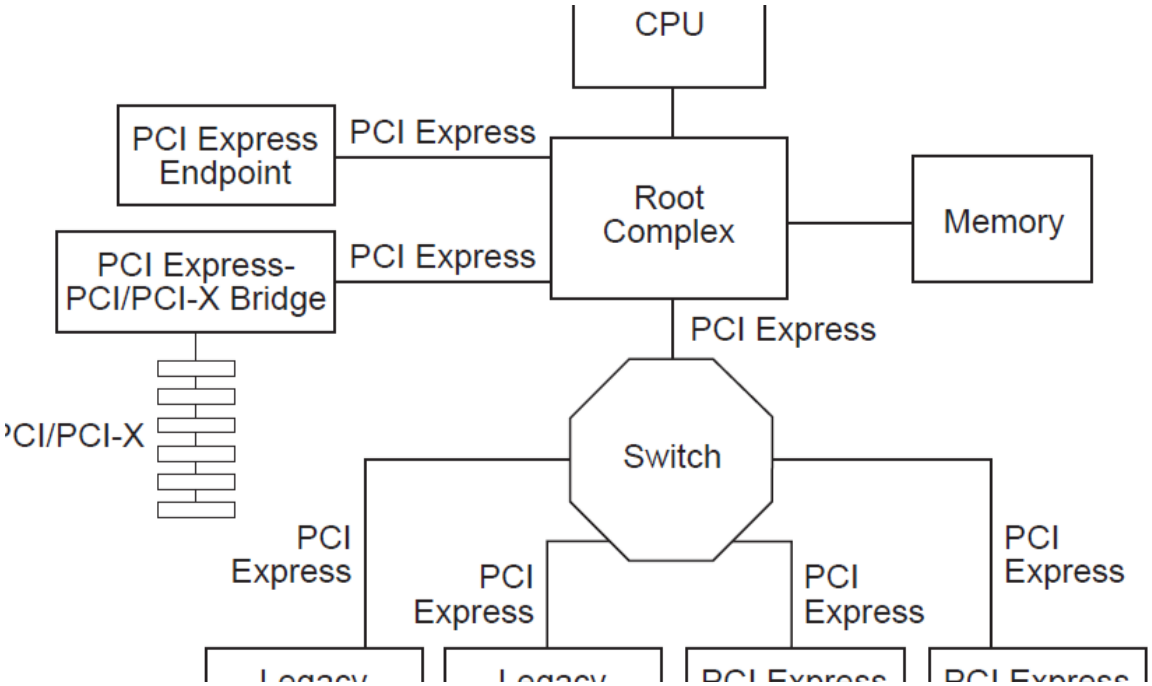
Key Benefits of Steering Tag in MSI-X

· **Optimized CPU Interrupt Handling** – Helps **steer interrupts** to the **most appropriate core**, reducing latency.

· **Improved Load Balancing** – Distributes **interrupts dynamically** across processing units, enhancing performance.

· **Better Performance in Multi-Threaded Environments** – Particularly **beneficial for networking, GPUs, and storage controllers** handling high interrupt loads

PCIe Gen 5 Key Features:



High-Speed Data Rate:

o **32.0 GT/s per lane**, doubling PCIe Gen 4 speeds, while maintaining backward compatibility with Gen 4 (16 GT/s), Gen 3 (8 GT/s), and earlier specifications.

Physical Layer Enhancements:

o **Lower PIPE interface pin count** at 32.0 GT/s, simplifying hardware design.

o **New PHY SerDes architecture** (per PIPE Specification 5.0/5.1) for improved high-speed signaling efficiency.

o **Configurable lane widths**:

Ø PIPE interface: 8, 16, 32, 64 bits.

Ø SerDes architecture: 10, 20, 40, 80 bits (scalable for performance/power trade-offs).

Link Optimization:

o **Optional skip of equalization** at lower data rates when operating at 32.0 GT/s, reducing link training overhead.

o Allows bypassing of the detailed equalization process if conditions are met, speeding up link initialization while potentially saving power.

o **Modified TS1/TS2 ordered sets** for enhanced link training and compatibility.

Power Management:

- Supports **low-power LTSSM states** (L0s, L1, L2, L1 substates) and standards like **ASPM** and **PCI-PM** for energy efficiency.
- L0s, L1, L2: Different levels of power-saving with varying latency for reactivation.
- L1 Sub-states: Further refinement for power management.
- PCI-PM: Traditional PCI power management states.
- ASPM: Active State Power Management for dynamic power adjustments during active use.
- **Latency Tolerance Reporting (LTR)** to optimize system power policies.

Advanced Error Handling:

- **Advanced Error Reporting (AER)** with optional features:
  - Malformed TLP checks.
  - End-to-End CRC (ECRC).
  - TLP Poisoning support.

Virtualization & Scalability:

- **SR-IOV (Single Root I/O Virtualization)** for partitioning devices into virtual functions.
- **Address Translation Services (ATS 1.1)** to reduce translation latency in virtualized environments.

Performance & Control Features:

- **10-bit Tag support** (requester/completer) for tracking more concurrent transactions.
- **Scaled flow control** and **lane margining** (signal integrity testing via voltage adjustments).
- **Function Level Reset (FLR)** for resetting individual functions without disrupting others.

Compliance:

- Backward-compliant with PCIe 4.0, 3.0, 2.0, and 1.1.
- Adheres to **PIPE Specification 5.1** (and 5.0 for SerDes architecture).

PCIe Gen 6.0 Features:

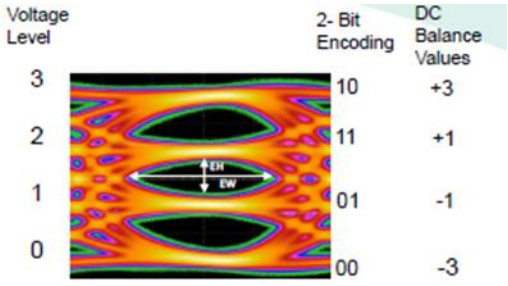
1.Advanced Data Transfer and Scalability

- **Description**: PCIe Gen 6 delivers 64 GT/s per lane, scaling to 256 GB/s bidirectional in x16 configurations and supporting up to 128 lanes for extensive interconnects, ideal for data centers and AI systems.
- **Technical Details**: The scalability stems from PAM-4 and FLIT encoding, which optimize bandwidth efficiency. A single x16 link offers 128 GB/s per direction, and 128 lanes could theoretically reach 1 TB/s unidirectional, though practical designs typically max at x16 or x32.

2. PAM-4 Signaling

- **Description**: PCIe Gen 6 shifts from the Non-Return to Zero (NRZ) signaling used in prior generations (e.g., PCIe 5.0 and earlier) to Pulse Amplitude Modulation with four levels (PAM-4). Unlike NRZ, which uses two voltage levels (0 and 1) to encode one bit per clock cycle, PAM-4 employs four voltage levels (representing 00, 01, 10, 11) to encode two bits per clock cycle.

- PAM-4 signaling: Pulse Amplitude Modulation 4-level
  - 4 levels (2 bits) encoded in same Unit Interval (UI)
  - 3 eyes
  - Helps channel loss (same Nyquist as 32.0 GT/s)
- Reduced voltage levels (EH) and eye width increases susceptibility to errors – 3 eyes in same UI
- Gray Coding to help minimize errors in UI
- Precoding to minimize errors in a burst
- Voltage levels at Tx and Rx define encoding



Encoding per UI (2bit)	Tx Voltage	Rx Voltage (V)
00	-Vtx	V <= Vth1
01	-Vtx/3	Vth1 < V <= Vth2
11	+Vtx/3	Vth2 < V <= Vth3
10	+Vtx	V > Vth3

3. Enhanced Encoding Mechanisms (FLIT Encoding)

- **Description**: PCIe Gen 6 introduces FLIT (Flow Control Unit) encoding with fixed 256-byte packets, replacing variable-size TLPs and eliminating 128b/130b encoding overhead. This enhances efficiency and supports PAM-4 and FEC.
- **Technical Details**: Each FLIT includes 236 bytes for TLP payload, 6 bytes for DLLPs (Data Link Layer Packets), and FEC/CRC overhead, yielding >90% TLP efficiency. Fixed sizes simplify error correction and data management.

Low-Latency with High Efficiency

- FLIT (flow control unit) based: FEC needs fixed set of bytes
- Correction in FLIT => CRC (detection) in FLITs => Retry at FLIT level
- Lower data rates will also use the same FLIT once enabled

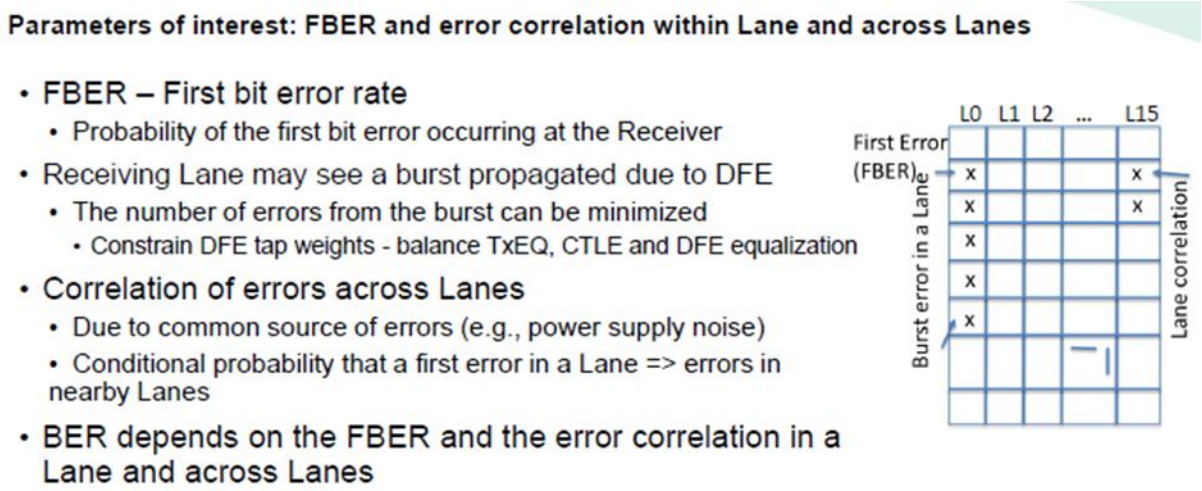
- FLIT size: 256B
  - 236B TLP, 6B DLP, 8B CRC, 6B FEC
  - No Sync hdr, no Framing Token (TLP reformat), no T(DL)LP CRC
  - Improved bandwidth utilization due to overhead amortization
  - FLIT Latency: 2ns x16, 4ns x8, 8 ns x4, 16 ns x2, 32 ns x1
  - Guaranteed Ack and credit exchange => low latency, low storage

- Optimization: Retry error FLIT only with existing Go-Back-N retry

x8 Lanes	0	1	2	3	4	5	6	7
256 UI								
TLP Bytes	0	1	2	3	4	5	6	7
(0-299)	8	9	10	11	12	13	14	15
	16	17	18	19	20	21	22	23
	24	25	26	27	28	29	30	31
	32	33	34	35	36	37	38	39
	40	41	42	43	44	45	46	47
	48	49	50	51	52	53	54	55
	56	57	58	59	60	61	62	63
	64	65	66	67	68	69	70	71
	72	73	74	75	76	77	78	79
	80	81	82	83	84	85	86	87
	88	89	90	91	92	93	94	95
	96	97	98	99	100	101	102	103
	104	105	106	107	108	109	110	111
	112	113	114	115	116	117	118	119
	120	121	122	123	124	125	126	127
	128	129	130	131	132	133	134	135
	136	137	138	139	140	141	142	143
	144	145	146	147	148	149	150	151
	152	153	154	155	156	157	158	159
	160	161	162	163	164	165	166	167
	168	169	170	171	172	173	174	175
	176	177	178	179	180	181	182	183
	184	185	186	187	188	189	190	191
	192	193	194	195	196	197	198	199
	200	201	202	203	204	205	206	207
	208	209	210	211	212	213	214	215
	216	217	218	219	220	221	222	223
	224	225	226	227	228	229	230	231
	232	233	234	235	dlp0	dlp1	dlp2	dlp3
	dlp4	dlp5	crc0	crc1	crc2	crc3	crc4	crc5
	crc6	crc7	ecc0	ecc0	ecc0	ecc1	ecc1	ecc1

4. Forward Error Correction (FEC)

- **Description**: PCIe Gen 6 introduces Forward Error Correction (FEC) to address the higher BER introduced by PAM-4 signaling. FEC corrects errors in real-time without retransmission, targeting a latency overhead of less than 2 ns, and works alongside Cyclic Redundancy Check (CRC) to minimize link retries.
- **Technical Details**: The lightweight FEC mechanism operates on fixed-size FLITs (256 bytes), adding error correction codes that the receiver uses to reconstruct corrupted data. This ensures reliability at 64 GT/s, where PAM-4's tighter signal margins increase error susceptibility.



5. Improved Power Efficiency

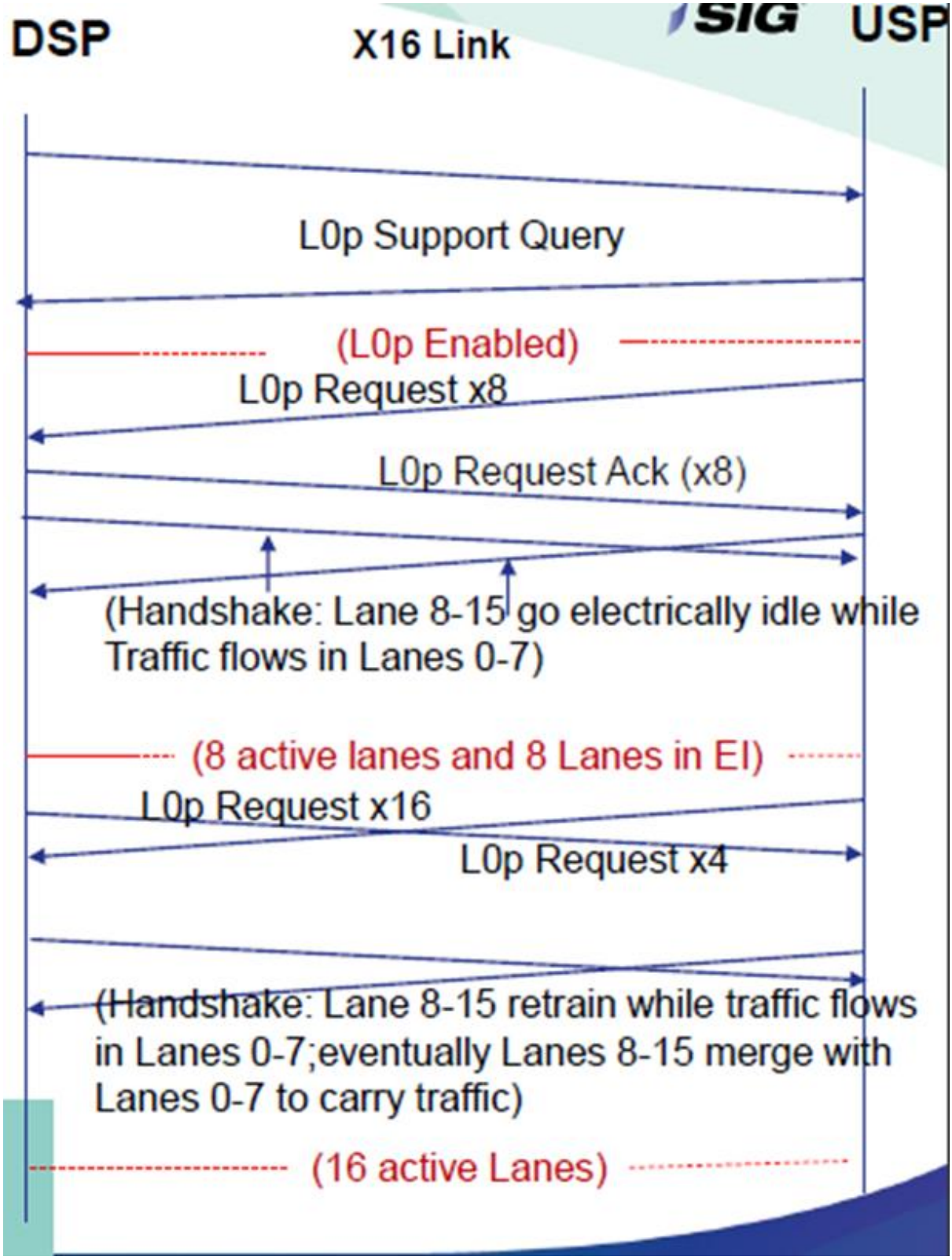
- **Description**: PCIe Gen 6 enhances power efficiency with the L0p low-power state, allowing dynamic lane scaling (e.g., reducing from x16 to x1) while maintaining at least one active lane for uninterrupted data flow. This reduces power consumption proportional to bandwidth usage.
- **Technical Details**: L0p replaces older low-power states (e.g., L0s) in FLIT mode, requiring links to train at maximum width (e.g., x16) and modulate down as needed. This is critical for data centers aiming to lower energy costs.

- Existing low-power states: L0s, L1, Dynamic Link Width (DLW), Speed Change
  - Served well for the set of usages so far and will continue

- Increasingly there is demand for ppwer consumption scaling with bandwidth usage without impacting traffic flow

- Solution: New state L0p – symmetric
  - Maintain at least one active Lane – they continue to carry traffic. Link still carries traffic during L0p width transition
  - Expect L0p PHY power savings similar to turning off power for the idle Lanes

L0p enables power consumption proportionate to bandwidth usage without interrupting traffic flow



6. Security Enhancements

- **Description**: PCIe Gen 6 adds Component Measurement and Authentication (CMA) and Integrity and Data Encryption (IDE). CMA verifies device authenticity via cryptographic signatures, while IDE encrypts FLIT packets to prevent tampering or interception.

- **Technical Details**: CMA uses firmware-based signatures, checked during initialization. IDE operates at the packet level (Link IDE for one-hop encryption, Selective IDE for multi-hop), requiring tight integration with controllers for low-latency encryption at 64 GT/s.

Key Matrix for PCIe gen6:

Metrics	Expectations	Evaluation (Trend)
Data Rate	64 GT/s, PAM4 (double the bandwidth per pin every generation)	Meets (must do)
Latency	<10ns adder for Transmitter + Receiver over 32.0 GT/s (including FEC) (We can not afford the 100ns FEC latency as n/w does with PAM-4)	Exceeds (Savings in latency with <10ns for x1/ x2 cases)
Bandwidth Inefficiency	<2 % adder over PCIe 5.0 across all payload sizes	Exceeds (getting >2X bandwidth in most cases)
Reliability	0 < FIT << 1 for a x16 (FIT – Failure in Time, failures in 10^9 hours)	Meets
Channel Reach	Similar to PCIe 5.0 specification under similar set up for Retimer(s) (maximum 2)	Meets
Power Efficiency	Better than PCIe 5.0 specification	Design dependent – expected to meet
Low Power	Similar entry/ exit latency for L1 low-power state Addition of a new power state (L0p) to support scalable power consumption with bandwidth usage without interrupting traffic	Design dependent – expected to meet; L0p looks promising
Plug and Play	Fully backwards compatible with PCIe 1.x through PCIe 5.0	Meets
Others	HVM-ready, cost-effective, scalable to hundreds of Lanes in a platform	Expected to Meet

PCIe Gen7 spec : Draft with PCI-SIG

PCI-SIG this week released version 0.7 of the PCI-Express 7.0 specification to its members. PCIe 7.0 is the next generation interconnect technology for computers that is set to increase data transfer speeds to 128 GT/s per pin, doubling the 64 GT/s of PCIe 6.0.

To achieve its impressive data transfer rates, PCIe 7.0 doubles the bus frequency at the physical layer compared to PCIe 5.0 and 6.0. Otherwise, the standard retains pulse amplitude modulation with four level signaling (PAM4), 1b/1b FLIT mode encoding, and the forward error correction (FEC) technologies that are already used for PCIe 6.0. Otherwise, PCI-SIG says that the PCIe 7.0 specification also focuses on enhanced channel parameters and reach as well as improved power efficiency.

PCI Express® 7.0 Specification & Status

PCIe 7.0 specification, version 0.3 is now live for PCI-SIG members; The full PCIe® 7.0 specification is targeted for release in 2025

- **What does Version 0.3 mean?**
  - The first review draft of the specification is complete and has received work group approval

Feature Goals:

- Delivering 128 GT/s data rate and up to 512 GB/s bi-directionally via x16 configuration
- Utilizing PAM4 signaling
- Defining the channel parameters
- Continuing to deliver the low-latency and high-reliability targets
- Improving power efficiency
- Maintaining backwards compatibility with all previous generations of PCIe technology

Revision	Max Data Rate	Encoding	Signaling
PCIe 7.0 (2025)	128.0 GT/s	1b/1b (Flit Mode*)	PAM4
PCIe 6.0 (2022)	64.0 GT/s	1b/1b (Flit Mode*)	PAM4
PCIe 5.0 (2019)	32.0 GT/s	128b/130b	NRZ
PCIe 4.0 (2017)	16.0 GT/s	128b/130b	NRZ
PCIe 3.0 (2010)	8.0 GT/s	128b/130b	NRZ
PCIe 2.0 (2007)	5.0 GT/s	8b/10b	NRZ
PCIe 1.0 (2003)	2.5 GT/s	8b/10b	NRZ

(*Flit Mode also enabled in other Data Rate with their respective encoding)

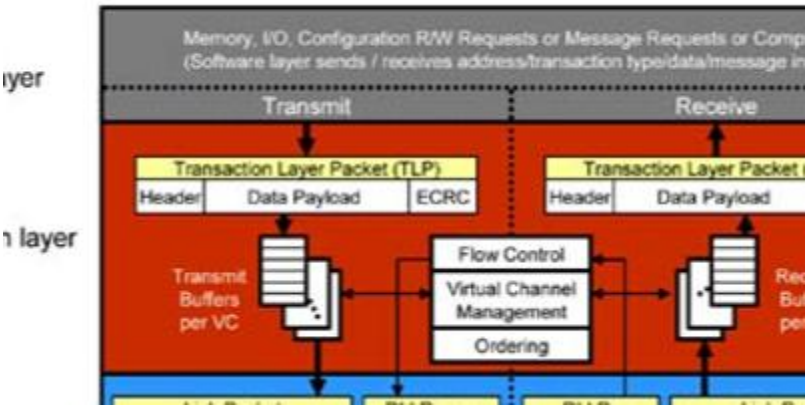
PCI Gen7 Specifications Status, Source: PCI-SIG

PCI Gen7 Specifications Status, Source: PCI-SIG

Past of the PCI, PCI-X, and PCI-Express speed jumps have looked like, and how we can roughly project out with a three-year cadence for specifications:

PCI	Spec	Data				Maximum Server Slot	
Specification	Released	Rate	Encoding	Signaling	Frequency	Bandwidth	Type
PCI	1992	1.06 Gb/sec	*32b/34b*	NRZ	33 MHz	133 MB/sec	32-bit Simplex
PCI 2.0	1993	4.26 Gb/sec	*64b/66b*	NRZ	66 MHz	533 MB/sec	64-bit Simplex
PCI-X	1999	8.5 Gb/sec	*64b/66b*	NRZ	133 MHz	1.06 GB/sec	64-bit Simplex
PCI-X 2.0	2002	17 Gb/sec	64b/66b	NRZ	266 MHz	2.13 GB/sec	64-bit Simplex
PCI-Express 1.X	2003	2.5 Gb/sec	8b/10b	NRZ	2.5 GT/sec	8 GB/sec	x16 Duplex
PCI-Express 2.X	2007	5 Gb/sec	8b/10b	NRZ	5 GT/sec	16 GB/sec	x16 Duplex
PCI-Express 3.X	2010	8 Gb/sec	128b/130b	NRZ	8 GT/sec	32 GB/sec	x16 Duplex
PCI-Express 4.0	2017	16 Gb/sec	128b/130b	NRZ	16 GT/sec	64 GB/sec	x16 Duplex
PCI-Express 5.0	2019	32 Gb/sec	128b/130b	NRZ	32 GT/sec	128 GB/sec	x16 Duplex
PCI-Express 6.0	2022	64 Gb/sec	1b/1b FLIT	PAM-4	64 GT/sec	256 GB/sec	x16 Duplex
PCI-Express 7.0	2025	128 Gb/sec	1b/1b FLIT	PAM-4	128 GT/sec	512 GB/sec	x16 Duplex
PCI-Express 8.0	*2028*	*256 Gb/sec*	1b/1b FLIT	*PAM-16*	*256 GT/sec*	*1 TB/sec*	*x16 Duplex*
PCI-Express 9.0	*2031*	*512 Gb/sec*	*???*	*???*	*512 GT/sec*	*2 TB/sec*	*x16 Duplex*
PCI-Express 10.0	*2034*	*1 Tb/sec*	*???*	*???*	*1 TT/sec*	*4 TB/sec*	*x16 Duplex*

Transaction Layer in PCIe Part -1: Overview and TLP



The **Transaction Layer (TL)** in PCIe (Peripheral Component Interconnect Express) is the topmost layer responsible for managing transaction-level communication between devices. It handles packetization, flow control, error detection, and routing.

Overview of the Transaction Layer (TL)

The TL sits between the device's core logic (application layer) and the Data Link Layer (DLL). Its primary roles include:

1. **Packetization**: Assembling and disassembling **Transaction Layer Packets (TLPs)**.
2. **Data Exchange**: Translating application-layer read/write requests into TLPs for transmission.
3. **Error Detection**: Checking for **End-to-End CRC (ECRC)** errors.
4. **Flow Control**: Managing buffer credits to prevent overflow.
5. **Address Space Support**: Handling **Memory, I/O, Configuration, and Message** transactions.

Transaction Layer Packet (TLP) Structure

A TLP consists of:

1. **Header** (3 or 4 Double Words (DW)): **Format Field**: Determines header size and payload presence: **Bit 0**: 0 = 3DW header, 1 = 4DW header. **Bit 1**: 0 = No payload, 1 = Payload included. **Type Field**: Specifies TLP type (e.g., Memory Read, Configuration Write). **Address/Request Details**: Varies by TLP type (e.g., 32-bit vs. 64-bit memory addresses).
2. **Data Payload** (0–1024 DW, configurable).
3. **TLP Digest (Optional)**: ECRC for error detection.

Figure 1 :- TLP packet structure

Byte 0		Byte 1						Byte 2				Byte 3
Format	Type	R	TC	R	Attr	R	TH	TD	EP	Attr	AT	length
Byte 4		Byte 5						Byte 6				Byte 7
(Byte 4-6 depends on type of TLP)												Last DW BE / First DW BE
Byte 8		Byte 9						Byte 10				Byte 11
(Byte 8-11 depends on type of TLP)												
Byte 12		Byte 13						Byte 14				Byte 15
(Byte 12-15 depends on type of TLP(not necessary always))												

Figure 2 :- General TLP header structure

Types of TLPs

TLPs are categorized into three types based on transaction requirements:

1. Posted TLPs

- **No response required** (e.g., Memory Writes, Messages).
- **Examples**: **Memory Write TLP**: Transfers data to a memory address. **Message TLP**: Used for interrupts, power management, or vendor-specific commands.

2. Non-Posted TLPs

- **Require a Completion TLP** (response).
- **Examples**: **Memory Read**: Requests data from a memory address. **I/O Read/Write**: Accesses legacy I/O space (limited to 32-bit, often 16-bit). **Configuration Read/Write**: Accesses PCIe configuration space (registers).

3. Completion TLPs

- **Response to Non-Posted TLPs** (e.g., returning read data).
- **Fields**: **Completer ID**: Identifies the responding device. **Status**: Indicates success/failure (e.g., "Unsupported Request"). **Data Payload**: For read completions.

TLP Header Formats

Each TLP type has a unique header structure:

Memory Request TLP

- **3DW Header**: 32-bit memory address.
- **4DW Header**: 64-bit memory address.
- **Key Fields**: Address: Target memory location. Length: Data payload size (in DW). First/Last DW Byte Enable: Controls byte granularity.

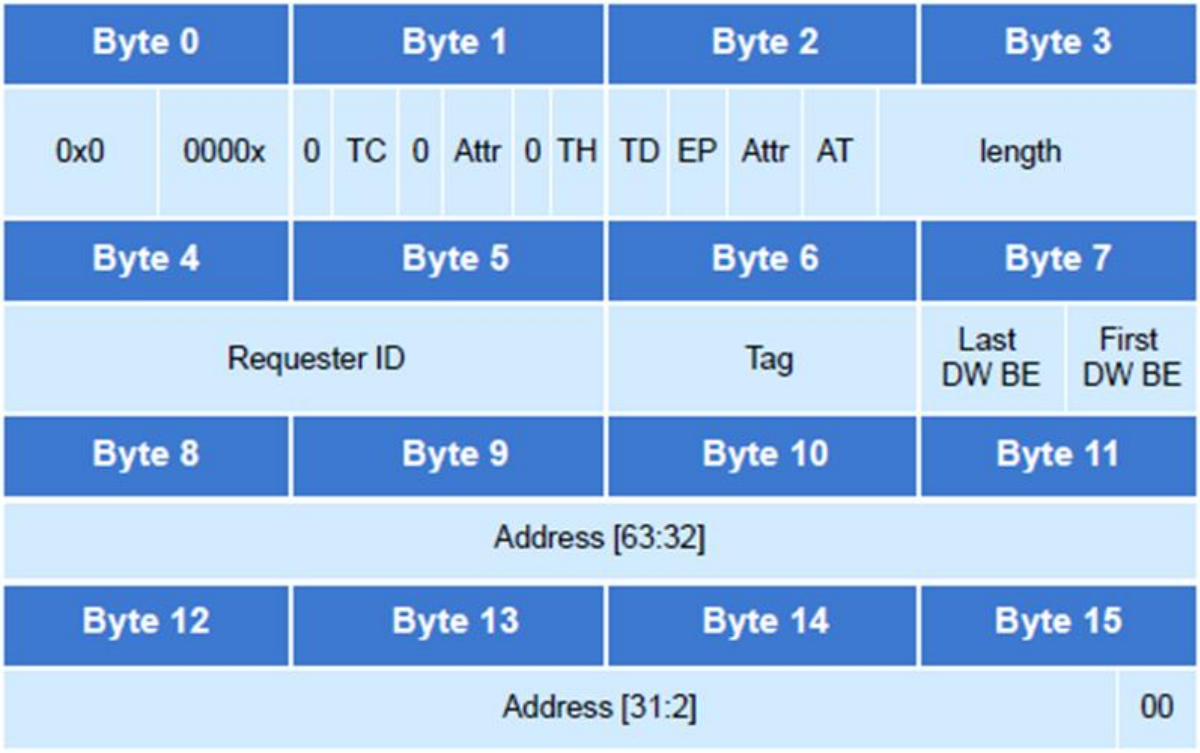


Figure 4 :- 4DW Memory request header

Configuration Request TLP

- Accesses PCIe configuration space (device registers).
- **Key Fields**: Bus/Device/Function (BDF): Identifies the target device. Register Number: Specifies the register offset.

Byte 0		Byte 1	Byte 2			Byte 3
0x0	0010x	00000000	TD	EP	0000	000000001
Byte 4		**Byte 5**	**Byte 6**			**Byte 7**
Requester ID			Tag		0000	First DW BE
Byte 8		**Byte 9**	**Byte 10**			**Byte 11**
Bus number		Device number / Function number	0000	Extended Register number	Register number	00

Figure 5 :- Configuration request header

I/O Request TLP

- Always uses a **4DW header** but supports only **1DW payload** (32-bit data).
- Limited to legacy systems (rarely used in modern PCIe).

Byte 0		Byte 1	Byte 2			Byte 3
0x0	00010	00000000	TD	EP	0000	000000001
Byte 4		**Byte 5**	**Byte 6**			**Byte 7**
Requester ID			Tag		0000	First DW BE
Byte 8		**Byte 9**	**Byte 10**			**Byte 11**
Address [31:2]						00

Figure 6 :- IO request header

Message Request TLP

- **4DW header** with a Message Code field (e.g., interrupt, error signaling).
- Bypasses traditional address spaces.

Byte 0		Byte 1					Byte 2			Byte 3
0x1	10xxx	0	TC	0	Attr	00	TD	EP	0000	length
Byte 4		**Byte 5**					**Byte 6**			**Byte 7**
Requester ID							Tag			Message code
Byte 8		**Byte 9**					**Byte 10**			**Byte 11**
Address [63:32]										
Byte 12		**Byte 13**					**Byte 14**			**Byte 15**
Address [31:2]										00

Figure 7 :- Message request header

Completion TLP

- Matches the original request using Requester ID and Tag.
- Includes Completion Status and data (for reads).

Byte 0		Byte 1					Byte 2			Byte 3
0x0	01010	0	TC	0	Attr	00	TD	EP	0000	000000001
Byte 4		**Byte 5**					**Byte 6**			**Byte 7**
Completer ID							Completion status	BCM		Byte Count
Byte 8		**Byte 9**					**Byte 10**			**Byte 11**
Requester ID							Tag		0	Lower address

Figure 8 :- Completion header

TLP Flow in PCIe

Transmitter:

- TL forms TLP (header, payload, ECRC if enabled).
- Passes to DLL, which adds sequence number and LCRC.
- PL adds STP (start), END, and serializes data.

Receiver:

- PL deserializes, removes STP/END.
- DLL verifies LCRC, sequence number, and sends TLP to TL.
- TL checks ECRC, extracts data, and delivers to core logic.

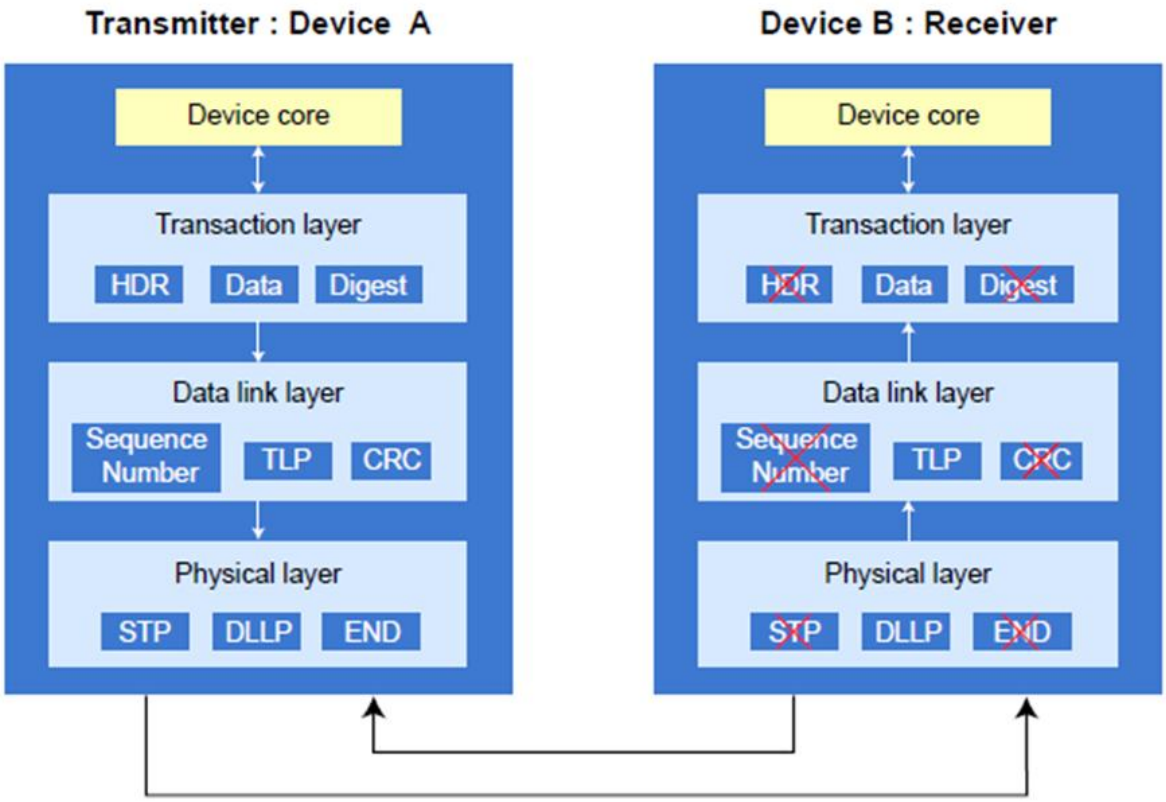


Figure 9 :- Data transmission in PCIe

Flow Control in the Transaction Layer

- **Credit-Based Mechanism**: The receiver reports buffer availability (**credits**) via **Data Link Layer Packets (DLLPs)**. Virtual Channels (VCs): Up to 8 independent channels for prioritizing traffic (e.g., high-priority vs. bulk data).
- **Process**:
- Transmitter checks receiver credits.

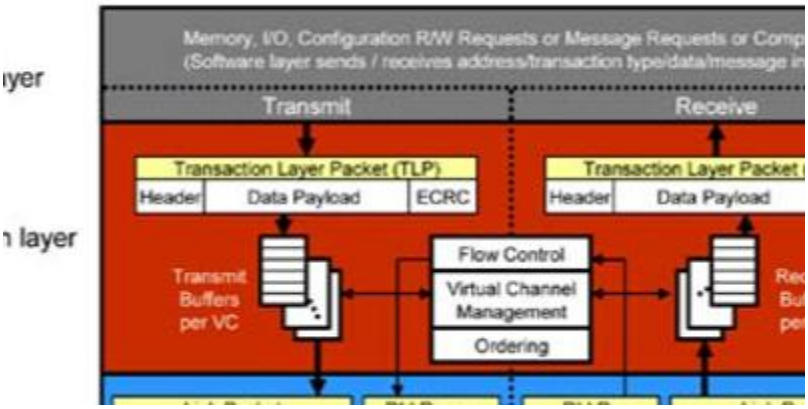
- Credits are updated via DLLPs exchanged between DLL layers.
- TLPs are sent only if credits are available.

Generic Header Fields:

- **Format** (Fmt): header size is 3 DW or 4 DW, payload presence.
- **Type**: Encodes the transaction type.
- **TC** (Traffic Class): Priority level (0–7) for QoS.
- **Attr** (Attributes): Controls ordering (Relaxed, ID-Based) and cache behavior (No Snoop).
- **TH** (TLP Processing Hints): 1 bit.
- **TD** (TLP Digest): 1 bit (ECRC presence).
- **EP** (Poisoned Data): 1 bit: Marks corrupted data (e.g., for error handling).
- **Length**: 10 bits (payload size in DW).
- **Requester ID**: Bus/Device/Function (BDF) of the transaction initiator.
- **Tag**: Unique ID for tracking non-posted transactions.
- **Completion ID**: Bus/Device/Function (BDF) of the transaction completer.
- **Transaction ID**: The combination of the Requester ID and the Tag field of the TLP is termed as Transaction ID of the TLP.

Transaction Layer in PCIe Part -2: Request and Completion Handling

---



Unsupported Requests (UR)

A Request is flagged as **Unsupported** (UR) if:

- **Request Type** is not supported by the device (e.g., unsupported Memory/I/O/Configuration command).
- **Message Request** uses an undefined or unsupported **Message Code** or **routing field** (except Vendor-Defined Type 1).
- **Message Routing by ID** targets an **unimplemented Function** (Bus/Device number mismatch).

Handling:

- **Non-Posted Requests** (e.g., Read): Return a **Completion with UR status**.
- **Posted Requests** (e.g., Write): Log UR as an error (no completion).

Message Requests

- **Supported Messages**: Processed per protocol rules (e.g., interrupts, power management).
- **Ignored Messages**: Silently discarded (no error reported).
- **Routing Rules**: **Routed by ID**: Type 0 devices (Endpoints) must act as targets, even if Bus/Device number mismatches. **Invalid Routing**: Treated as UR (e.g., broadcast messages at Downstream Ports).

Completer Abort (CA)

A **Completer Abort** is triggered if:

- The device encounters a **permanent error** (e.g., hardware failure).

- The Request violates the device's **programming model** (e.g., unaligned access to a register).

Handling:

- **Non-Posted Requests**: Return a **Completion with CA status**.
- **Posted Requests**: Log CA as an error.
- **Note**: CA is **not** for requests that would have been ignored in PCI (use UR instead).

Configuration Request Retry Status (CRS)

- **Usage**: Returned for **Configuration Requests** after resets (Cold/Warm/Hot/FLR) if the device is initializing.
- **Conditions**: CRS is **only valid post-reset** (not after software resets). Device must not use CRS after signaling **Configuration-Ready**.
- **Root Complex Handling**: Re-issue the Request until success or synthesize 0001h for Vendor ID reads if CRS Software Visibility is enabled.

Completion Handling

- **General Rules**: **Successful Completions**: Return data (if applicable) with status SC. **ECRC Errors**: Optional Completion with UR status (recommended).
- **Read Completions**: Must respect **Max_Payload_Size** and **Read Completion Boundary (RCB)** (64/128 bytes). Multi-Completions must return data in **contiguous address order**.

Timing Constraints

Posted Request Acceptance Limit

- **Rule**: Devices must accept **Posted Requests** (e.g., Memory Writes) within **10µs** under normal operation.
- **Exceptions**: Post-reset periods, link retraining, diagnostic modes. Delays from arbitration or low-power state transitions.
- **Mitigation**: Use a **restricted programming model** (e.g., driver polls buffer availability).

Non-Posted Requests

- **I/O Writes**: Completions should ideally adhere to the same 10µs limit (recommended, not enforced).

Special Cases

- **FLR (Function-Level Reset)**: Requests received during FLR may be silently discarded.
- **Legacy Devices**: Must handle all legacy-compatible Requests to avoid software incompatibility.
- **PCI/PCI-X Bridges**: Translate PCI-side errors (Master/Target Abort) to UR/CA in PCIe.

Key Error Reporting

Error Type	Trigger	Completion Status
Unsupported (UR)	Invalid Request Type/Message/Address.	UR
Completer Abort (CA)	Permanent device error or protocol violation.	CA
CRS	Configuration Request during device reset.	CRS

Byte Enables in PCIe Transactions

Specify which bytes within a Data Word (DW) are valid for read/write operations.

- **First DW Byte Enables (BE[3:0])**: Control bytes in the first DW.
- **Last DW Byte Enables (BE[7:4])**: Control bytes in the last DW (for multi-DW requests).

Key Scenarios

Example 1: Single-DW Write

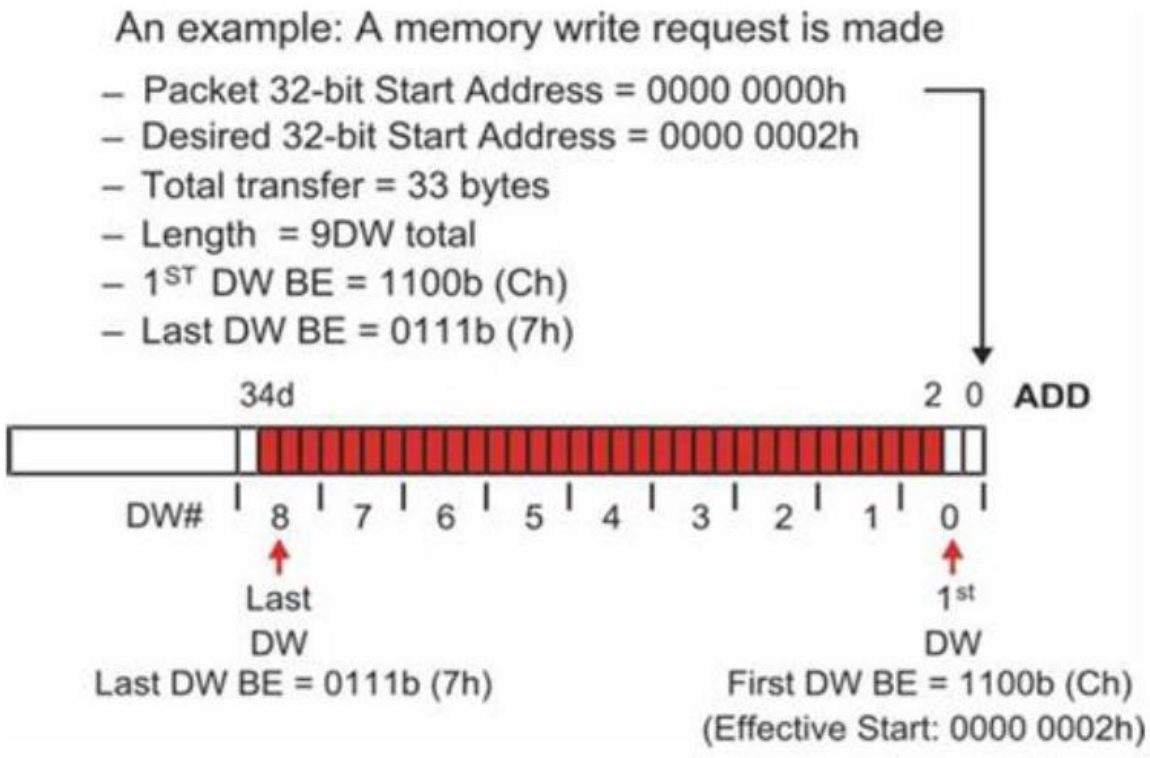
- **First DW BE**: 1010b → Write bytes 0 and 2.
- **Last DW BE**: 0000b (required).

Example 2: Multi-DW Read (QW-Aligned)

- **First DW BE**: 1100b (bytes 3-2).
- **Last DW BE**: 0011b (bytes 1-0).
- **Contiguous Data**: Bytes 3-2 (first DW) + bytes 1-0 (last DW) = 4-byte contiguous block.

Example 3: Invalid Case

- **Multi-DW Request** with First DW BE = 0000b → Violates rule (rejected as Malformed TLP).



Data Returned for Read Requests

· **Multiple completions**: A read request may require multiple completions to fulfill, but the total data transfer must equal the size of the original request, or a **Completion Timeout** error may occur.

· **Single completion**: A completion can only service **one request** at a time.

· **IO and Configuration reads**: These are always 1 DW in size and are completed with a single completion.

· **Unsuccessful completions**: A completion with a status other than **SC (Successful Completion)** terminates the transaction.

Read Completion Boundary (RCB)

· The **RCB** is either **64 bytes** or **128 bytes**, depending on the configuration of the **Root Complex**. It represents the boundary within which completions must occur.

· **RCB Alignment**: Completions that fit entirely within an aligned RCB boundary must complete in one transfer. If the transaction crosses an RCB, it must stop at the RCB boundary.

Flow chart for TLP Handling:

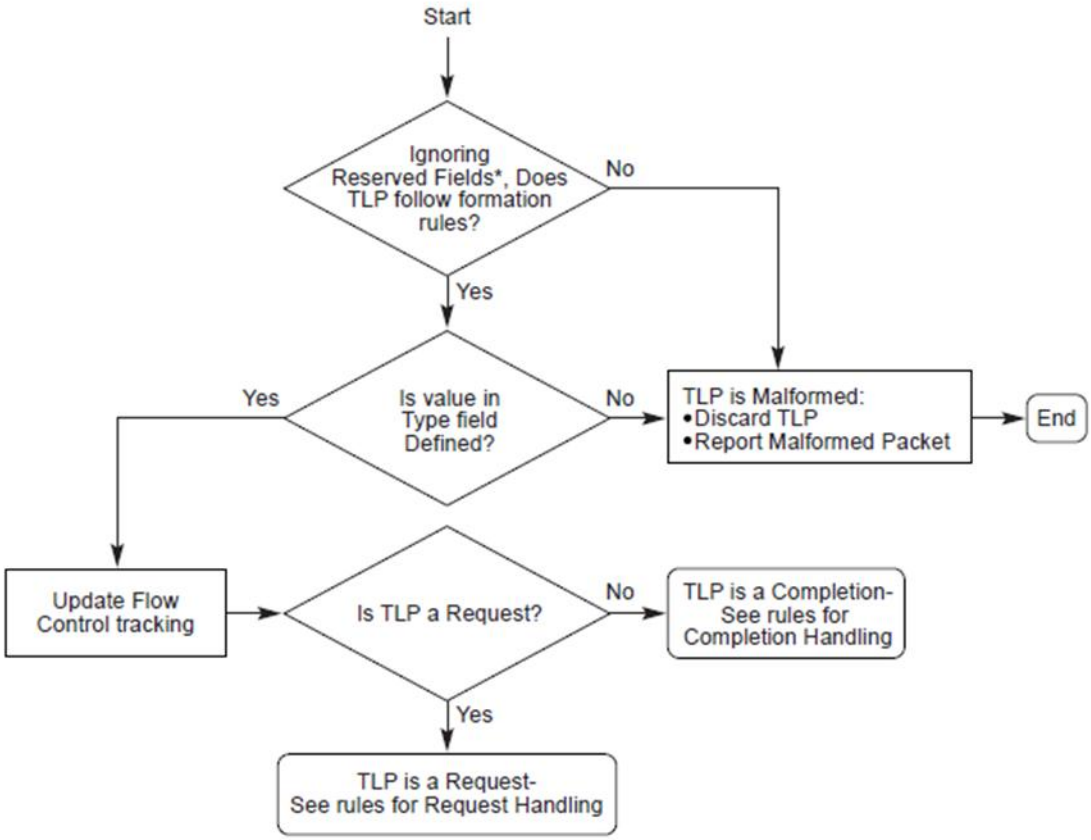


Figure 2-41 Flowchart for Handling of Received TLPs

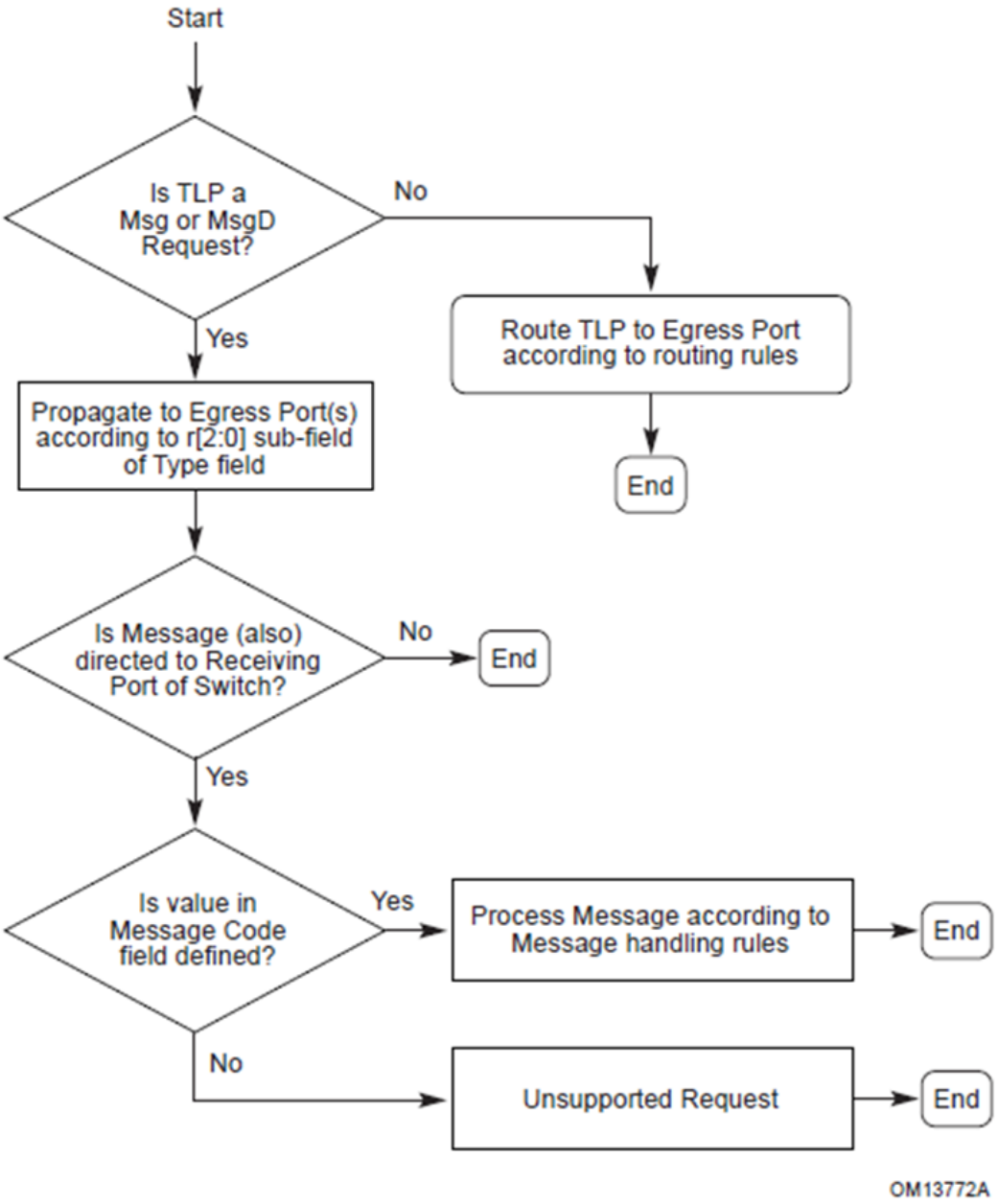


Figure 2-42 Flowchart for Switch Handling of TLPs

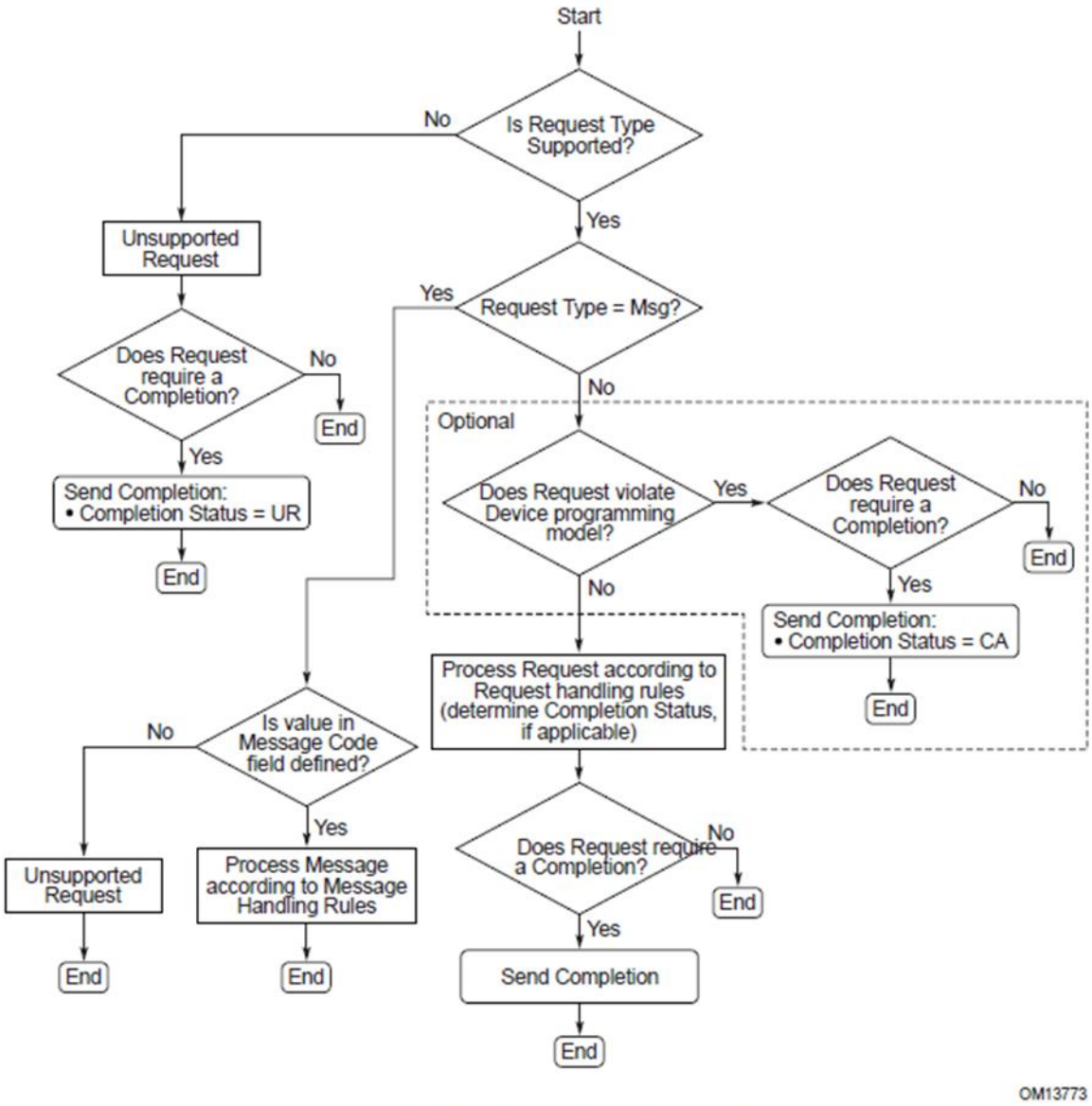
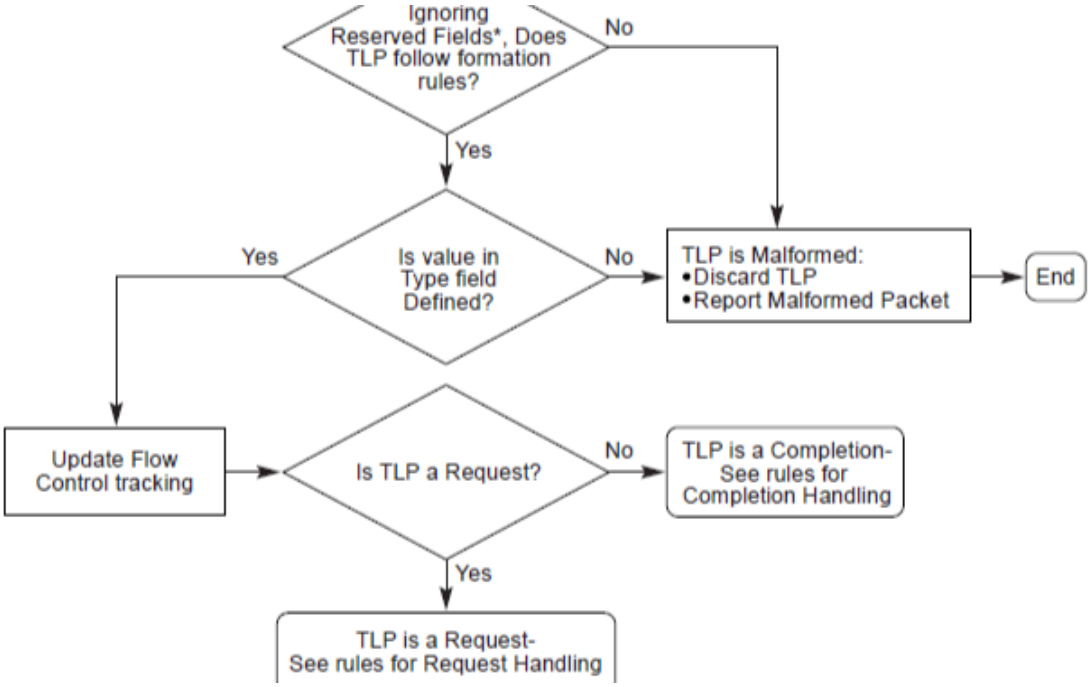


Figure 2-43 Flowchart for Handling of Received Request

Transaction Layer in PCIe Part - 3: Message TLP



Message TLPs are posted requests, meaning they do not require a Completion TLP response, and they facilitate a variety of system-level functions. Their primary uses include:

Interrupt Signaling:

1. **INTx Messages**: Emulate legacy PCI interrupts (e.g., INTA, INTB), sent as Assert_INTx or Deassert_INTx to signal interrupt state changes.
2. **MSI (Message Signaled Interrupts)**: Deliver interrupt requests by writing to a specific memory address, identified by the Message Code (e.g., MSI or MSI-X).

Power Management:

- **PM Messages**: Manage device power states, e.g., PM_Active_State_Nak (rejects a power state change), PM_PME (Power Management Event, signals wake-up), or PM_Enter_L1 (requests low-power state).

Power Management Message Types

Power Management Message	Message Code 7:0	Routing 2:0
PM_Active_State_Nak	0001 0100b	100b
PM_PME	0001 1000b	000b
PM_Turn_Off	0001 1001b	011b
PME_TO_Ack	0001 1011b	101b

Error Reporting: Error Messages: Signal errors to the Root Complex:

- ERR_COR: Correctable error (e.g., CRC error).
- ERR_NONFATAL: Non-fatal uncorrectable error (e.g., poisoned TLP).
- ERR_FATAL: Fatal uncorrectable error (e.g., link failure).

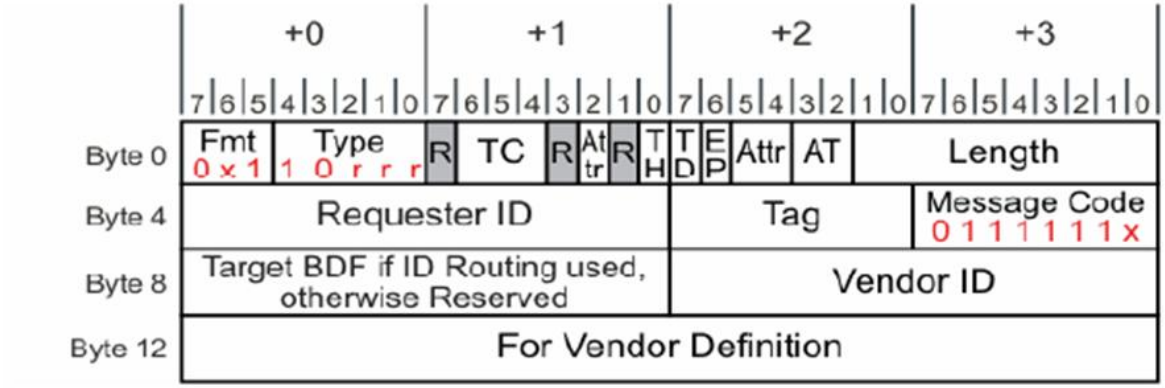
Locked Transaction Support:

- **Unlock Message**: Signals completion of a locked transaction sequence.

Vendor-Defined Messages:

- Allow custom functionality between devices, using Vendor_Defined Type 0 or Type 1 Message Codes.

Vendor-Defined Message Header



Vendor-Defined Message Coding

Vendor-Defined Message	Message Code 7:0	Routing 2:0
Vendor Defined Message 0	0111 1110b	000b, 010b, 011b, 100b
Vendor Defined Message 1	0111 1111b	

Slot Power Control:

- Manage power limits for slots (e.g., Set_Slot_Power_Limit).

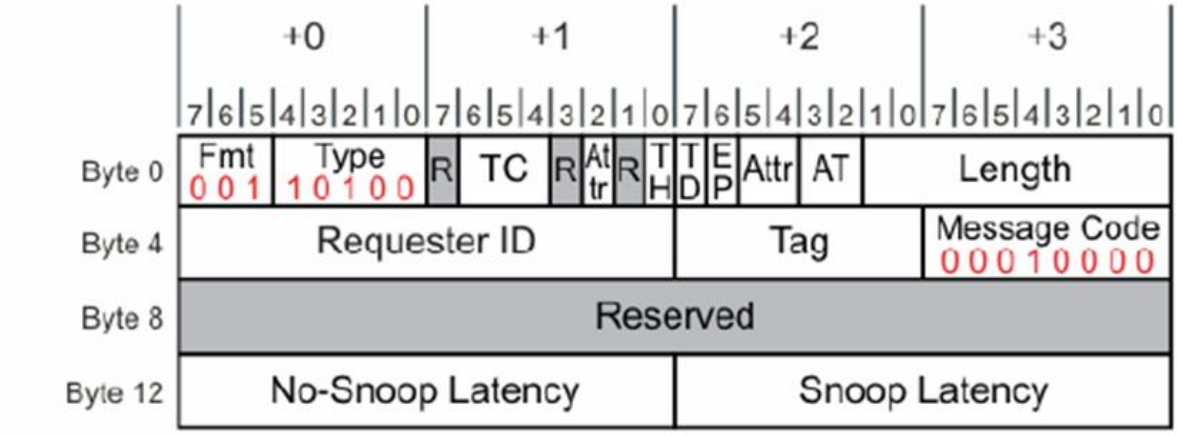
Hot-Plug and Miscellaneous Events:

- Signal events like device insertion/removal or latency tolerance updates (e.g., Latency_Tolerance_Reporting).

Latency Tolerance Reporting (LTR) Message

· Allows a PCIe device to report its acceptable service latencies for memory transactions to the system, enabling optimized power management by balancing performance and energy usage.

Key Points of LTR Messages



Optimized Buffer Flush and Fill (OBFF) Message

· Enables the platform (e.g., Root Complex) to inform PCIe endpoints of its power status, allowing endpoints to optimize buffer usage and power consumption based on system activity.

Key Characteristics

- **Header**: Always 4 DW (Double Words, 16 bytes), with no data payload (Msg) or with payload (MsgD), indicated by Format field (Fmt[2:0] = 100b for no data, 101b for data).

- **Routing**: Uses implicit, address, or ID-based routing, specified by the r[2:0] subfield in Type.
- **Traffic Class**: Typically uses TC0 (default), with violations treated as Malformed TLPs.

Handling of Message TLPs

Handling Message TLPs at the PCIe Transaction Layer involves validating, routing, and processing them, as outlined in "Request Handling Rules" and related sections. Since they are posted, no Completion TLP is generated, and handling focuses on error detection and action.

1. **Initial Validation**:

o **Malformed Check**: Invalid Fmt/Type (e.g., reserved values), TC ≠ 0 (if required), or payload mismatch with Length.

Action: Discard as Malformed TLP, report error (Section 6.2), no flow control update if ambiguous.

2. **Result**: Valid TLPs proceed.

3. **Routing Determination**:

o **Implicit Routing** (r[2:0] specific values):

i. 000b: To Root Complex (RC), forwarded upstream by switches.

ii. 011b: Broadcast from RC, forwarded to all downstream ports by switches.

ii. 100b: Local (terminates at receiver).

4. **Address Routing** (r[2:0] = 001b): Uses Address field, routed like Memory Writes.

o **ID Routing** (r[2:0] = 010b): Uses Bus, Device, Function numbers.

o **Switch Handling**:

i. Upstream port rejects 000b messages (Malformed TLP).

ii. Downstream port rejects 011b messages (Malformed TLP).

5. **Support Check**:

o **Condition**: Verify if the Message Code and routing are supported by the device.

o **Unsupported**:

i. Treated as Unsupported Request (UR), discarded, reported per Section 6.2.

ii. Exception: Vendor_Defined Type 1 not treated as UR.

6. **Supported**: Proceed to process.

o **Ignored Messages**: If configured to ignore (e.g., per Section 2.2.8.7), discard silently without error.

7. **Processing**:

o **Action**: Depends on Message Code:

i. Assert_INTx: Signal interrupt to system.

ii. PM_PME: Wake device, notify RC.

iii. ERR_NONFATAL: Log error, notify RC.

8. **Payload**: If present (e.g., MsgD), process data (e.g., power limit value in Set_Slot_Power_Limit).

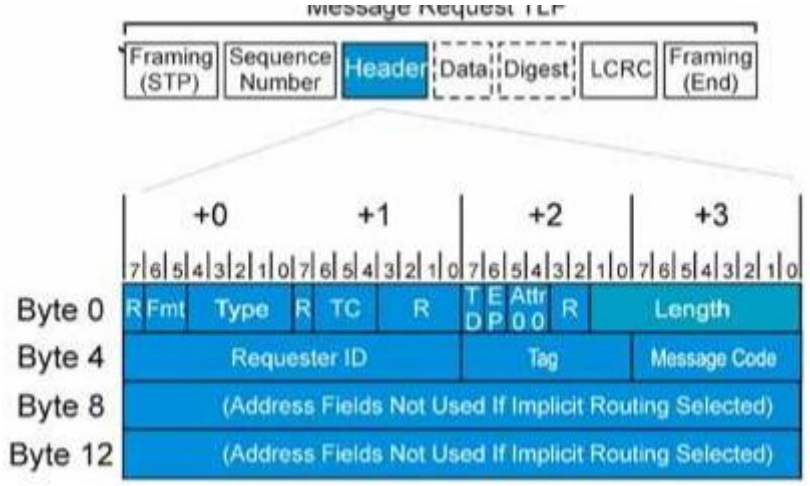
9. **Flow Control Update**:

o Credits updated (Section 2.6) for valid TLPs, ensuring buffer space tracking.

Error Handling

- **UR**: Discarded, error logged (Section 6.2), no further action (posted nature).
- **Malformed TLP**: Discarded, reported, may skip flow control update if ambiguous.

Transaction Layer in PCIe Part - 4: Misc.



Alternative Routing-ID Interpretation**:**

· ARI is an optional feature that increases the number of functions a single device can support from 8 to 256.

· It achieves this by reinterpreting the Device Number and Function Number fields as a single 8-bit Function Number.

· ARI benefits both virtualized and non-virtualized environments by allowing more concurrent users of a multi-Function device.

· Requires new software and is managed via the ARI Capability structure in ARI Devices and the Device Capabilities 2 and Device Control 2 registers in ARI Downstream Ports.

· **Backward Compatibility**: ARI is designed to be backward compatible with existing PCI Express specifications.

· **Extended Functions**: Functions with a Function Number greater than 7 are Extended Functions, accessible after ARI Forwarding is enabled in the Downstream Port above the ARI Device.

· **Function Groups**: Functions within an ARI Device can be configured into Function Groups for VC arbitration or access control.

· **Topology**: Root Ports and Switch Downstream Ports must support ARI Forwarding to access Extended Functions.

Traditional Multi-Function Devices (non-ARI):

· Limited to a single Device Number but can implement up to eight independent Functions.

· Each internal Function is selected based on decoded address information provided as part of the address portion of Configuration Request packets.

· **Phantom Functions Supported Field**: In non-ARI multi-Function devices, the Phantom Functions Supported field within each Function's Device Capabilities register indicates support for using unclaimed Function Numbers to extend the number of outstanding transactions. This is achieved by logically combining unclaimed Function Numbers (called Phantom Functions) with the Tag identifier. **ARI Devices and Phantom Functions**: With every Function in an ARI Device, the Phantom Functions Supported field must be set to 00. This indicates that Phantom Functions are not supported in ARI Devices

Flow Control Basics:

- **Mechanism**: FC uses FCPs (e.g., InitFC, UpdateFC DLLPs) to transfer credit information between transmitter and receiver.
- **Independence**: Each VC has its own FC credit pool, managed separately to prevent inter-VC blocking.
- **Credit Unit**: 4 Double Words (DW, 16 bytes) for data payload credits.

- Receivers supporting End-End TLP Prefixes must include the maximum prefix size in credit calculations.

2. TLP Types and Credit Tracking

- **Categories**: FC distinguishes three TLP types: **Posted Requests (P)**: E.g., Memory Writes, no completion required. **Non-Posted Requests (NP)**: E.g., Memory Reads, require completion. **Completions (Cpl)**: Responses to non-posted requests.
- **Subdivision**: Each type splits into Header (H) and Data (D), yielding six credit types per VC: PH: Posted Headers PD: Posted Data NPH: Non-Posted Headers NPD: Non-Posted Data CplH: Completion Headers CplD: Completion Data
- **Purpose**: Separate tracking prevents one type (e.g., PD) from consuming credits needed for another (e.g., NPH).

3. Credit Consumption

- **Calculation**: TLPs consume credits based on type and size: **Example**: Memory Write (posted): 1 PH credit for the header. n PD credits for data, where n = Roundup(Length / 4 DW). E.g., Length = 10 bytes → Roundup(10 / 16) = 1 PD credit.
- **Granularity**: Credits align to 4 DW units for data, ensuring efficient buffer allocation.

4. Virtual Channel Management

- **VC0**: Initialized autonomously by hardware post-reset (no software required).

- **Other VCs**: Enabled/disabled by software via VC Capability structures (Section 7.9.1), resetting FC tracking.
- **InitFC DLLPs**: **InitFC1**, **InitFC2**: Used during VC initialization to establish credit limits. Sent post-reset or when enabling a new VC.

5. Credit Limits and Errors

- **Limits**: Non-Scaled FC: Max 2047 data credits (PD, NPD, CplD) and 127 header credits (PH, NPH, CplH). **Flow Control Protocol Error (FCPE)**: Exceeding these limits triggers an error, reported per Section 6.2.
- **Infinite Credits**: If advertised during initialization (e.g., CREDIT_LIMIT = infinite), no further UpdateFC FCPs are needed for that type.

6. Virtual Channel Enablement

- **Disabled VC**: TLPs using a disabled VC are treated as Malformed TLPs, discarded, and reported (Section 6.2).
- **Transmission Rule**: No TLP transmission until VC initialization completes (credits established).

7. Transmitter Tracking

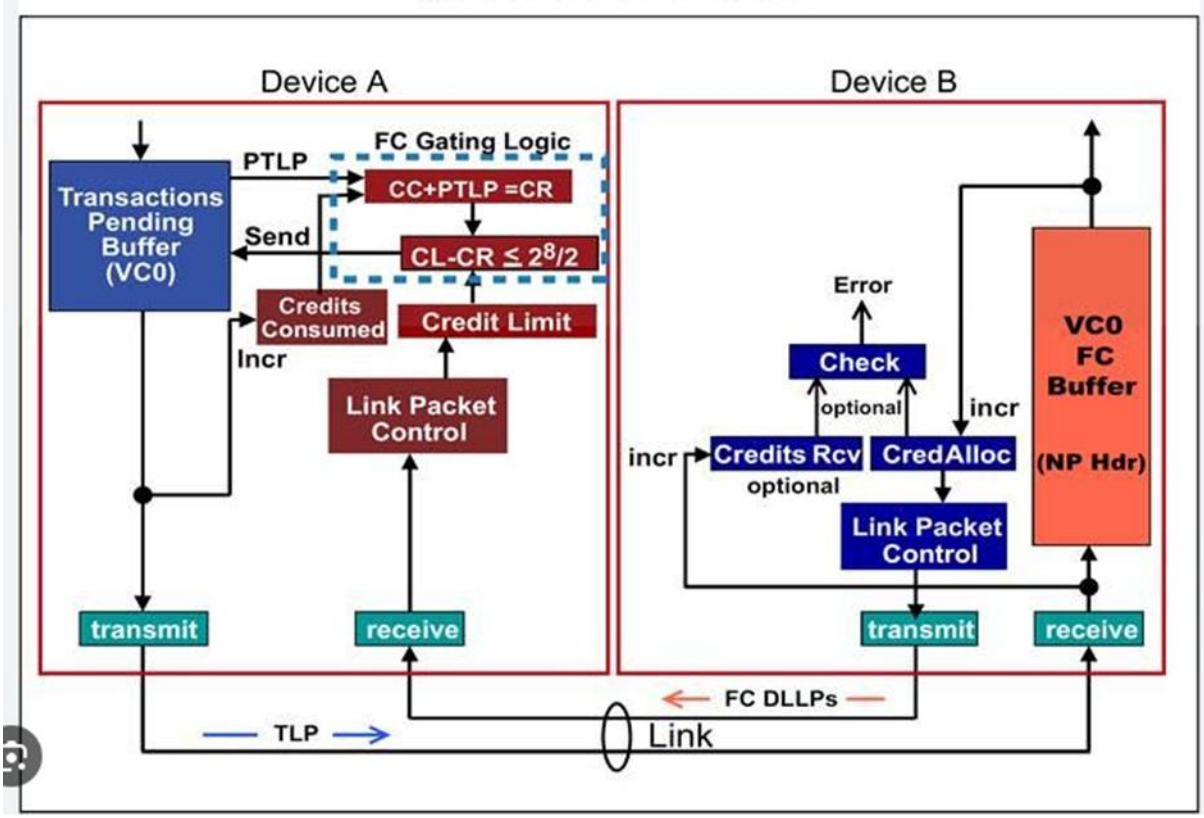
- **Variables**: **CREDITS_CONSUMED**: Tracks FC units consumed by transmitted TLPs (increments per TLP). **CREDIT_LIMIT**: Total credits advertised by the receiver (set via InitFC/UpdateFC).
- **Gating**: Transmission allowed if (CREDIT_LIMIT - CUMULATIVE_CREDITS_REQUIRED) mod 2^[Field Size] ≤ 2^[Field Size]/2. **Infinite Credits**: If CREDIT_LIMIT is infinite, gating is always satisfied (no credit check).
- **Deadlock Avoidance**: Ensures transmitter waits for sufficient credits, preventing buffer overflow.

8. Receiver Tracking

- **Variables**: **CREDITS_ALLOCATED**: Total credits granted to the transmitter, updated via InitFC/UpdateFC DLLPs. **CREDITS_RECEIVED** (optional): Tracks credits consumed by received TLPs.
- **Overflow Check**: Optional equation: (CREDITS_ALLOCATED - CREDITS_RECEIVED) mod 2^[Field Size] ≥ 2^[Field Size]/2.
- Detects if transmitter exceeds allocated credits.

9. UpdateFC FCP Scheduling

- **Conditions**: UpdateFC DLLPs are sent when: Available credits for a type (e.g., PH) reach zero. Scaled FC is active, and credits fall below a threshold (implementation-specific).
- **Purpose**: Ensures transmitter knows current buffer availability, maintaining flow.



Virtual Channel (VC) Mechanism

The Virtual Channel (VC) mechanism in PCIe enables the differentiation of traffic flows by associating them with independent virtual channels, each identified by a **Virtual Channel Identification (VC ID)**. This allows multiple traffic streams, labeled with **Traffic Classes (TCs)**, to coexist on the same physical link with separate flow control, preventing a single flow from blocking others. The VC mechanism is foundational to Quality of Service (QoS) and efficient resource utilization in PCIe systems.

Key Points

1. Purpose and Functionality

- **Traffic Differentiation**: VCs use TC labels to categorize traffic (e.g., TC0 for general I/O, TC7 for high-priority), enabling differentiated handling across the PCIe fabric.

- **Independent Resources**: Each VC has dedicated queues/buffers and control logic, ensuring fully independent flow control. This eliminates bottlenecks where one traffic flow could stall others (e.g., flow-control-induced blocking).

2. TC to VC Mapping

- **Mechanism**: Traffic is mapped to VCs using TC labels, controlled by configuration software.
- **Flexibility**: **1:1 Mapping**: One TC per VC (e.g., TC0/VC0, TC1/VC1). **Multiple TCs per VC**: Multiple TCs can share a VC (e.g., TC0-6/VC0, TC7/VC1) for cost/performance trade-offs.
- **Default Mapping**: TC0 is always mapped to VC0 (hardwired), mandatory for all devices.

3. VC Establishment and Identification

- **VC ID**: A unique identifier (0-7) assigned to each VC resource within a port, set by configuration software.
- **Support**: VC0 (with TC0) is mandatory; additional VCs (1-7) are optional. Ports supporting >1 VC must implement VC or MFVC (Multi-Function VC) Capability structures.
- **Rules**: VC IDs must be unique within a port and match across both sides of a link. VC0 is fixed as the default VC.

4. Interoperability

- **Non-VC Devices**: Devices without VC capability must: Generate requests only with TC0. Accept and preserve non-TC0 labels in requests/completions, mapping all TCs to VC0 (for switches).
- **VC-Capable Devices**: Must support configurable TC/VC mappings via capability structures.

5. Implementation Details

- **Buffering**: Minimum buffering is architecturally defined, but additional buffering is implementation-specific and may vary per VC (e.g., more for VC0 than VC1). Buffers can be reassigned dynamically if fewer VCs are enabled (e.g., VC1 buffers to VC0).
- **Switch/MFD**: Switches require dedicated VC resources per port. Multi-Function Devices (MFDs) may use MFVC for QoS across functions.
- **Configuration**: Software scans VC/MFVC Capability registers to set up VCs across links.

6. TC/VC Mapping Examples

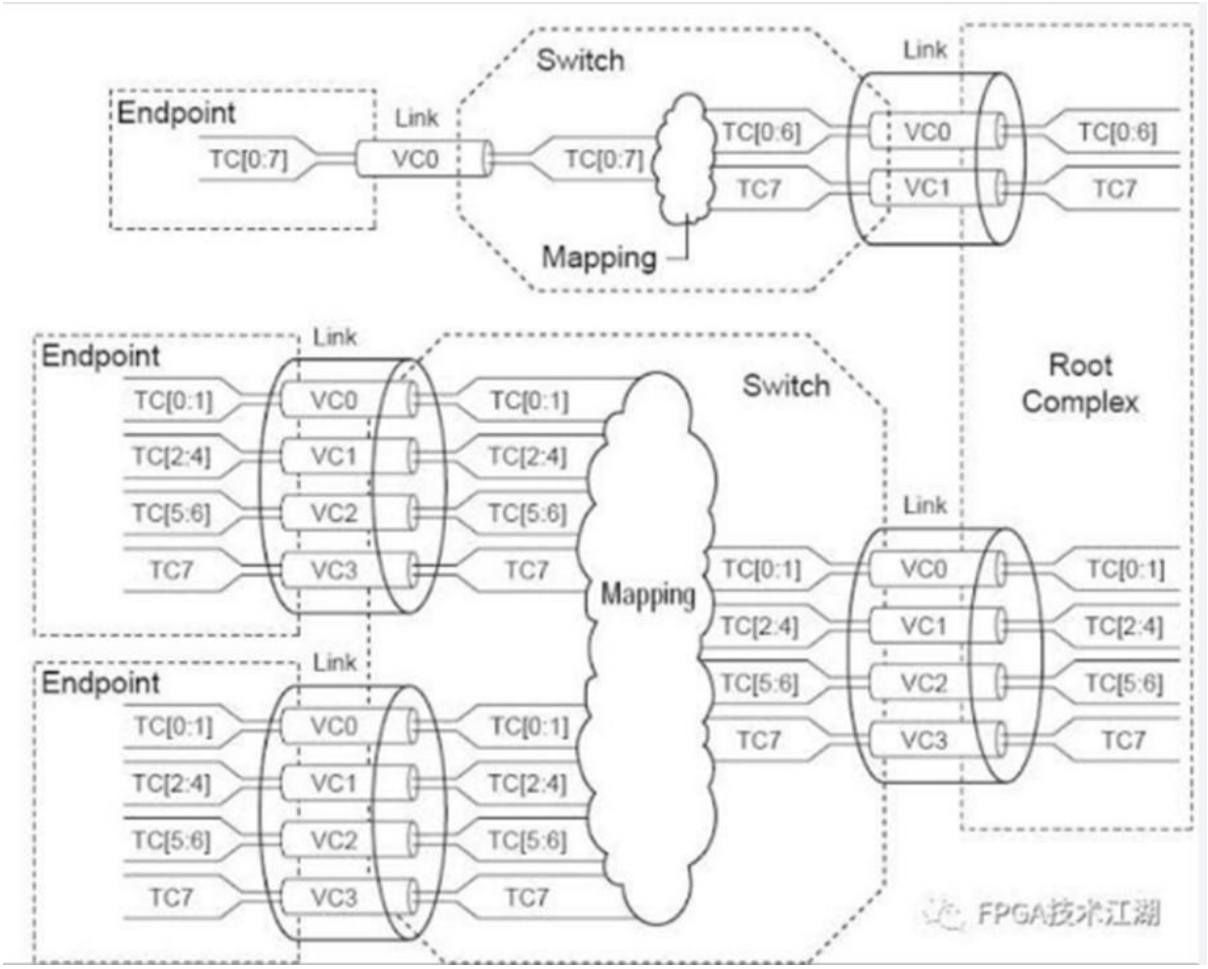
- **VC0**: TC0-7/VC0 (single VC).
- **VC0, VC1**: TC0-6/VC0, TC7/VC1 (two VCs).
- **VC0-VC3**: TC0-1/VC0, TC2-4/VC1, TC5-6/VC2, TC7/VC3 (four VCs).
- **VC0-VC7**: TC0/VC0, ..., TC7/VC7 (eight VCs, 1:1 mapping).

7. Rules and Constraints

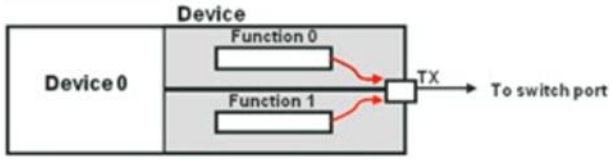
- **Mandatory Support**: TC0/VC0 is required for all devices.
- **Independent Flow Control**: Each VC has its own credit pool, managed via DLLPs (Data Link Layer Packets) with VC ID.
- **No Ordering**: No ordering required between different TCs or VCs, enhancing flexibility.
- **Malformed TLPs**: Transactions with unmapped TCs (to enabled VCs) are treated as Malformed TLPs and discarded.
- **Port Independence**: Switches and Root Complexes support independent TC/VC mappings per port/RCRB.

8. Flow Control Integration

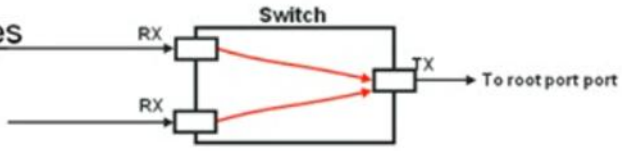
- **Point-to-Point**: Flow control is link-specific, not end-to-end, tracking buffer space between adjacent devices.
- **VC-Specific**: Each VC maintains independent credits, conveyed via DLLPs with VC ID, ensuring no inter-VC blocking.
- **TL Role**: Manages flow control credits, gating TLP transmission based on available credits.



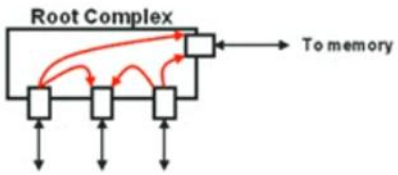
- Contention for usage of an egress port in PCIe can come from several sources
  - Multi function devices



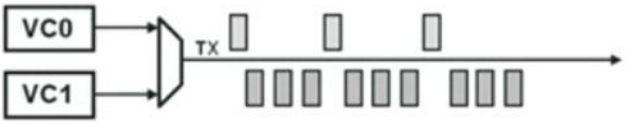
  - Switches



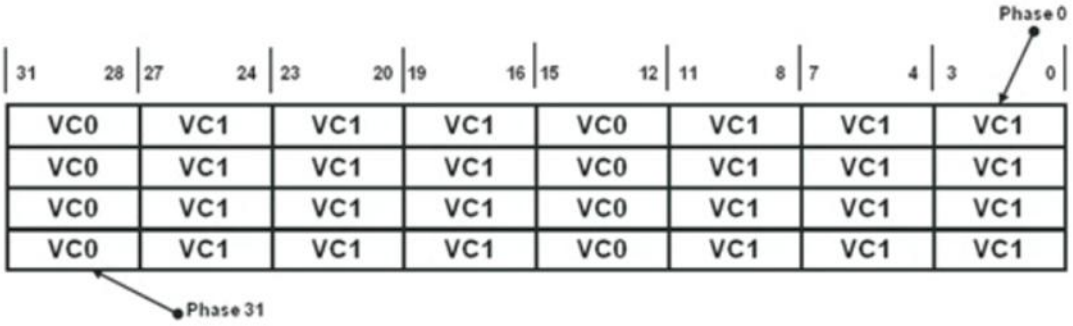
  - Within root complex



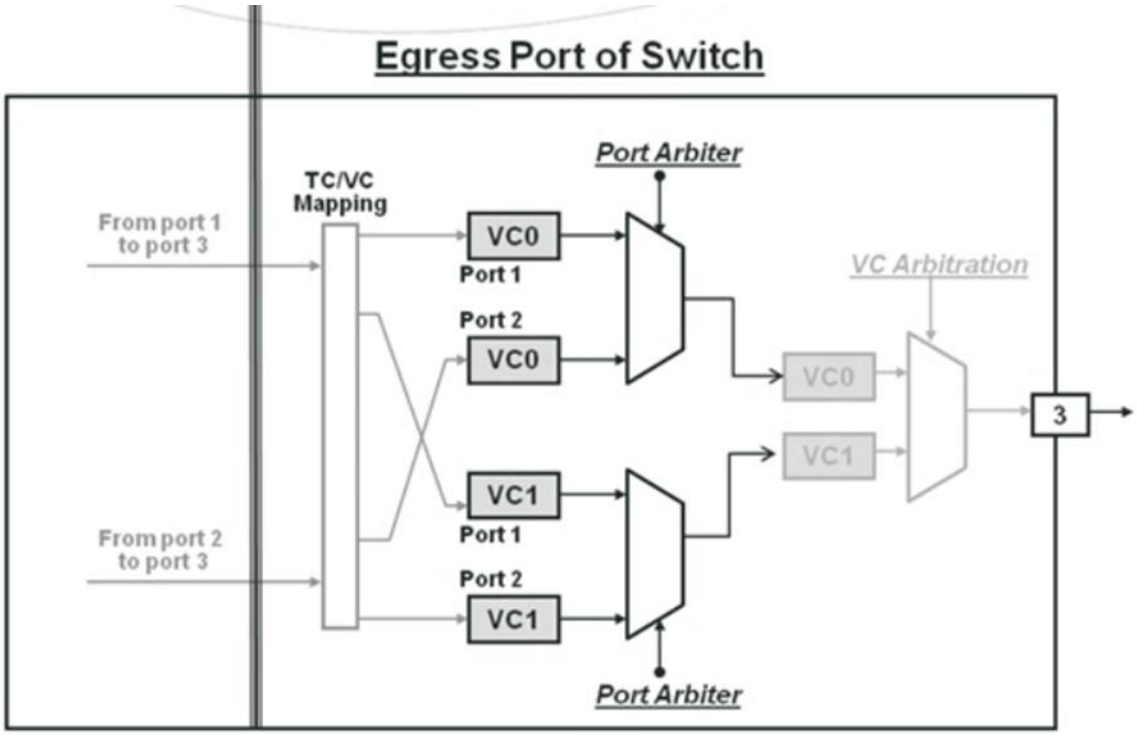
- Packets associated with VC1 are transmitted at 3 times the rate of packets with VC0



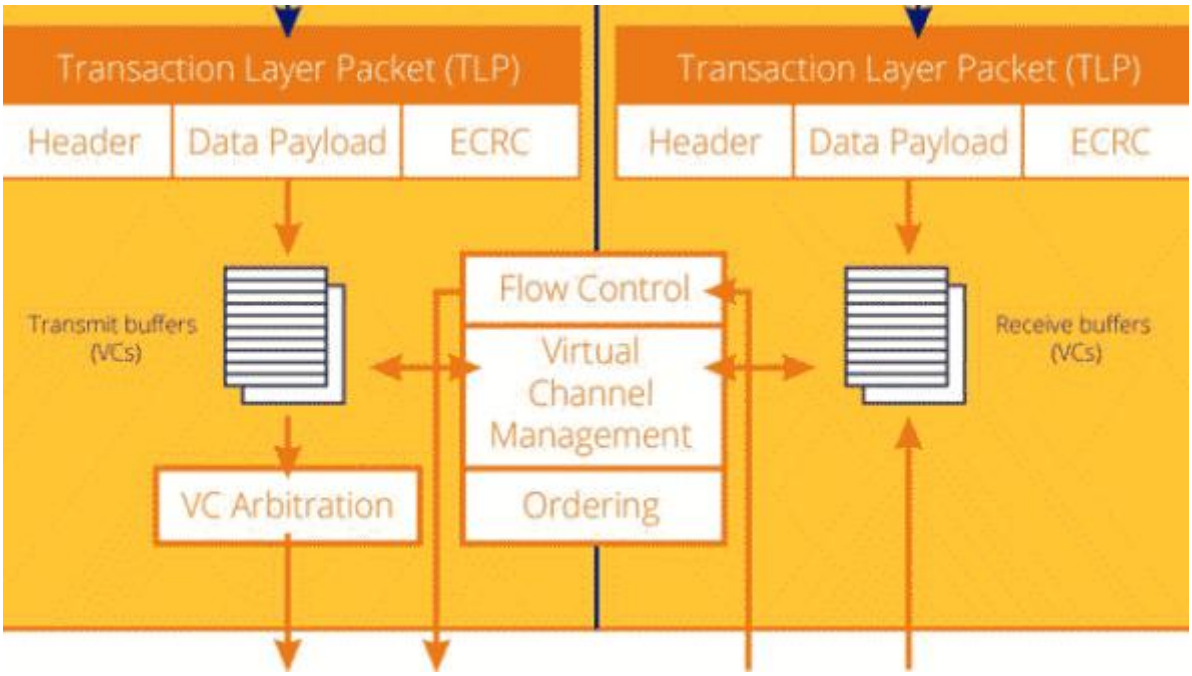
Virtual Arbitration Table (WWR with 32 phases)



Egress Port of Switch



Test Plan for PCIe Transaction Layer (TL)



Objective

To verify the PCIe Gen 5 Transaction Layer's ability to manage transaction-level communication, including TLP creation, routing, flow control, error detection, request/completion handling, Message TLPs, byte enable scenarios, ARI, and VC mechanisms, ensuring compliance with PCIe Base Specification Revision 5.0.

TL Specifications

- **Inputs**: From application layer (read/write requests) and DLL (TLPs).
- **Outputs**: To DLL (TLPs) and application layer (data, status).
- **Features**: TLP Types: Posted (Memory Write, Message), Non-Posted (Memory Read, I/O, Config), Completion. Bandwidth: 32 GT/s per lane, 128 GB/s (x16 bidirectional). Flow Control: Credit-based, up to 8 VCs. Error Handling: UR, CA, CRS, ECRC. Routing: Address, ID, Implicit (Message TLPs). Special Features: ARI (256 functions), Byte Enables, BCM, Message TLPs (LTR, OBFF).

Test Cases

1. Reset Behavior

- **Objective**: Verify TL resets correctly and initializes default state.
- **Stimulus**: Assert reset, send TLPs during/post-reset.
- **Checks**: No TLP processing during reset. VC0 credits initialized post-reset (autonomous). ARI disabled by default.

- **Coverage**: Reset with active transactions.

2. TLP Packetization and Structure

- **Objective**: Ensure correct TLP assembly/disassembly.
- **Stimulus**: Memory Write: 3DW/4DW header, 32/64-bit address, 64-byte payload, ECRC enabled. Config Read: BDF targeting Function 0, 4DW header. Message: Assert_INTA, 4DW header, no payload.
- **Checks**: Header: Correct Fmt (3DW/4DW, payload/no payload), Type, Length. Payload: Matches Length (0-1024 DW). ECRC: Present if TD=1, 32-bit CRC calculated.
- **Coverage**: All TLP types, header formats, ECRC presence/absence.

3. Posted TLP Handling

- **Objective**: Verify processing of posted TLPs (Memory Write, Message).
- **Stimulus**: Memory Write: Address = 0x1000, Length = 4 DW, First DW BE = 1010b. Message Variants: PM_PME, ERR_COR, Set_Slot_Power_Limit.
- **Checks**: Data written to memory (Memory Write). Message actions (e.g., wake-up for PM_PME, error log for ERR_COR). No Completion TLP returned.
- **Coverage**: All Message Codes, TC0-7, routing (implicit, address, ID).

4. Non-Posted TLP Handling

- **Objective**: Test non-posted requests and Completion generation.
- **Stimulus**: Memory Read: Address = 0x2000, Length = 8 DW. I/O Write: 32-bit address, 1 DW payload. Config Read: Valid BDF, register offset 0x10.
- **Checks**: Completion TLP returned: Status = SC, correct Byte Count, data for reads. Completer ID matches DUT. Credits updated.
- **Coverage**: Memory/I/O/Config, 32/64-bit addresses, payload sizes.

5. Completion TLP Handling

- **Objective**: Verify requester-side Completion processing.
- **Stimulus**: Valid: Tag matches request, Status = SC, Byte Count = 16. Unexpected: Unmatched Tag. Malformed: Incorrect Byte Count.
- **Checks**: Valid: Data accepted, resources freed. Unexpected: Discarded, error reported (Section 6.2). Malformed: Reported or treated as Unexpected.
- **Coverage**: SC, UR, CA, CRS status; multi-completion scenarios.

6. Byte Enable Scenarios

- **Objective**: Validate byte enable handling per "Key Scenarios."
- **Stimulus**: Single-DW Write: First DW BE = 1010b, Last DW BE = 0000b. Multi-DW Read: First DW BE = 1100b, Last DW BE = 0011b. Invalid: First DW BE = 0000b (multi-DW).
- **Checks**: Single: Bytes 0, 2 written. Multi: 4-byte contiguous block (bytes 3-2, 1-0). Invalid: Rejected as Malformed TLP.
- **Coverage**: Valid/invalid BE combinations, single/multi-DW.

7. Unsupported Request (UR) Handling

- **Objective**: Test UR conditions and handling.
- **Stimulus**: Unsupported Type: I/O Read on non-I/O device. Message: Undefined Msg Code = 0xFF. ID Routing: Unimplemented Function (e.g., Function 8 without ARI).
- **Checks**: Non-Posted: Completion with Status = UR. Posted: Discarded, error logged (Section 6.2).
- **Coverage**: UR for all TLP types, Vendor_Defined Type 1 exception.

8. Completer Abort (CA) Handling

- **Objective**: Verify CA for errors and violations.
- **Stimulus**: Unaligned Memory Read (e.g., Address = 0x1001). Hardware failure simulation.
- **Checks**: Non-Posted: Completion with Status = CA. Posted: Error logged.
- **Coverage**: CA for programming model violations, permanent errors.

9. Configuration Request Retry Status (CRS)

- **Objective**: Test CRS post-reset behavior.
- **Stimulus**: Config Read post-FLR, DUT not Configuration-Ready.
- **Checks**: Completion with Status = CRS. Root Complex retries or returns 0001h (Vendor ID) with CRS Visibility.
- **Coverage**: CRS post Cold/Warm/Hot/FLR resets, non-CRS scenarios.

10. BCM Bit Handling

- **Objective**: Verify BCM bit in Completion TLPs.
- **Stimulus**: Memory Read split by PCI-X bridge: First Completion: BCM = 1, Byte Count = 16. Second Completion: BCM = 0, Byte Count = 48.

- **Checks**: BCM = 1: Byte Count = first TLP size, more expected. BCM = 0: Completes remaining bytes.
- **Coverage**: BCM = 1/0, multi-Completion reads.

11. Message TLP Handling

- **Objective**: Test all Message TLP types and handling.
- **Stimulus**: Assert_INTA, PM_PME, ERR_NONFATAL, LTR, OBFF, Vendor_Defined Type 0. Invalid routing (e.g., r[2:0] = 011b at Downstream Port).
- **Checks**: Supported: Processed (e.g., interrupt signaled, power state updated). Ignored: Discarded silently (e.g., per Section 2.2.8.7). Unsupported: UR, logged.
- **Coverage**: All Message Codes, routing options (implicit, address, ID), TC0-7.

12. ARI Functionality

- **Objective**: Verify ARI support for Extended Functions.
- **Stimulus**: Config Read to Function 10 with ARI enabled/disabled. Function Group configuration (e.g., VC arbitration).
- **Checks**: ARI Enabled: Accesses Function 10, Phantom Functions Supported = 00b. ARI Disabled: Limited to Functions 0-7.
- **Coverage**: ARI on/off, Function Groups, non-ARI Phantom Functions.

13. Flow Control and VC Management

- **Objective**: Test credit-based flow control across VCs.
- **Stimulus**: VC0: Memory Write, 64 bytes. VC1-7: Enable via VC Capability, send TLPs with TC1-7. Credit exhaustion (e.g., exceed PD credits).
- **Checks**: VC0: Autonomous post-reset, credits updated. VC1-7: Enabled via software, independent credits, no inter-VC blocking. Gating: TLPs blocked if credits insufficient.
- **Coverage**: All VC IDs (0-7), TC mappings (1:1, multiple TCs/VC), credit limits (2047 data, 127 headers).

14. Timing Constraints

- **Objective**: Verify Posted Request Acceptance Limit.
- **Stimulus**: Back-to-back Memory Writes exceeding 10 µs.
- **Checks**: Accepted within 10 µs (normal conditions). Exceptions (post-reset, link retraining) handled.
- **Coverage**: Normal vs. exception conditions, I/O Write timing.

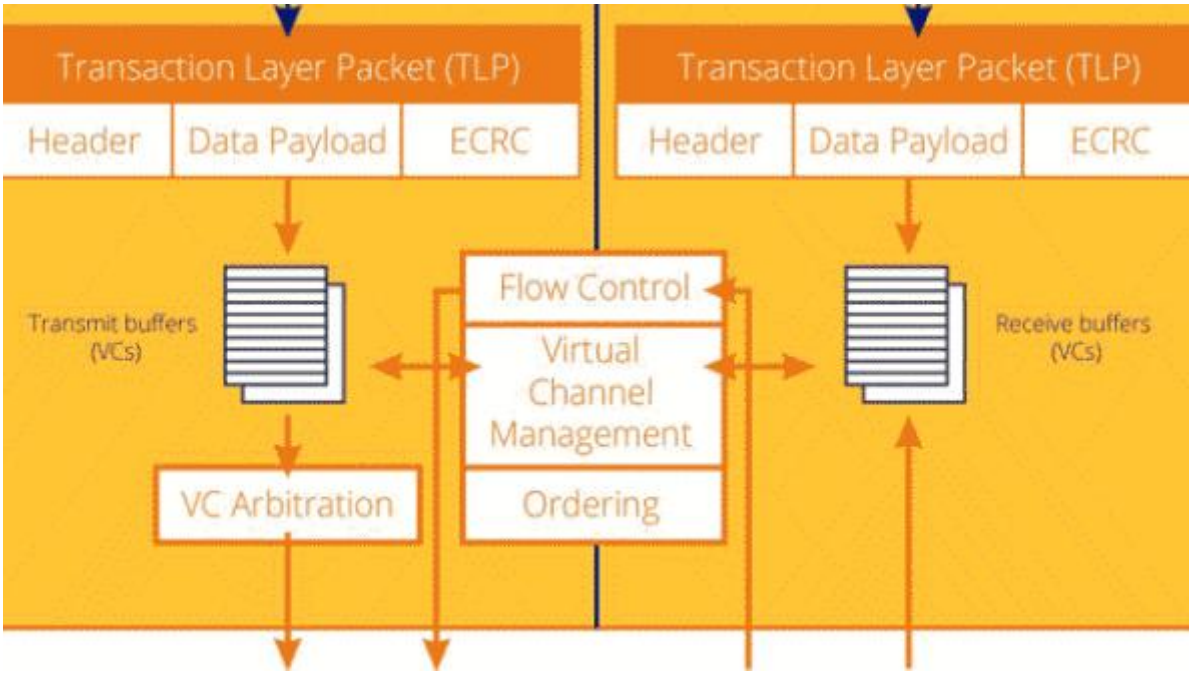
15. Stress Test

- **Objective**: Test TL at max bandwidth with mixed traffic.
- **Stimulus**: Concurrent Posted (Memory Write, Message), Non-Posted (Memory Read), Completion TLPs at 128 GB/s (x16).
- **Checks**: No data loss, correct error handling, VC prioritization.
- **Coverage**: Max throughput, all TLP types, VC0-7.

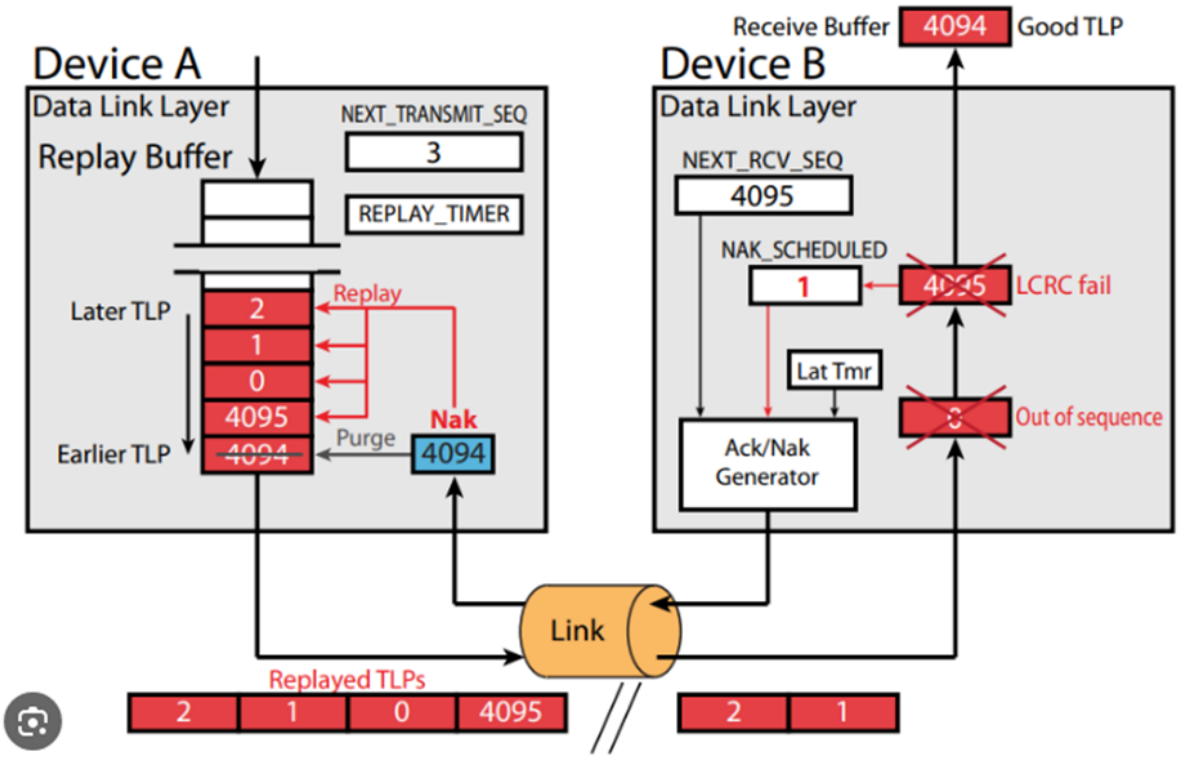
Verification Environment

- **Components**: **Agent**: Driver: Sends TLPs to DUT. Monitor: Captures DUT responses (TLPs, errors). Sequencer: Drives sequences. **Scoreboard**: Predicts Completions for non-posted TLPs. Verifies UR/CA/CRS status, Message TLP actions. **Virtual Sequencer**: Coordinates multi-VC, ARI, and priority sequences.
- **Assertions**: TLP header (Fmt, Type, Length) validity. ECRC match/mismatch handling. Credit consumption vs. limits. 10 µs posted limit compliance.
- **Coverage**: **Functional**: TLP types, VC IDs, TC mappings, ARI functions, Message Codes, error conditions (UR, CA, CRS).

PCIe Data link layer



The PCIe Gen 5 DLL serves as the intermediary between the Transaction Layer (TL) and Physical Layer (PL), ensuring reliable data transfer across the link. Its primary tasks include:



1. Link Management State Machine (DLCMSM) Supervision

- **Role**: Manages flow control initialization, tracks link status, and coordinates activity changes (e.g., power states, retraining).
- **Details**: Conveys link status (e.g., active, down) to TL and PL. Triggers actions like link retraining on failure.

2. Data Exchange

- **Transmit Path**: Receives TLPs from TL, adds a 12-bit **Sequence Number** and 32-bit **Link CRC (LCRC)**, then passes them to PL for encoding and transmission.
- **Receive Path**: Validates incoming TLPs (LCRC, Sequence Number), ensuring error-free delivery to TL.

3. Error Detection

- **Mechanism**: Uses LCRC to verify TLP integrity.
- **Actions**: Error-free TLPs pass to TL. Errors trigger NAK, link retraining, or retry mechanisms.

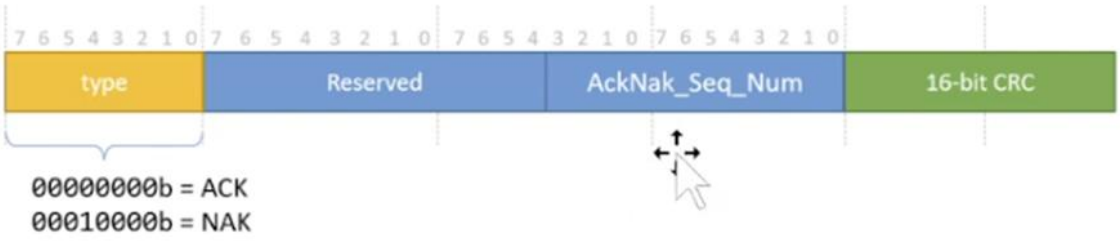
4. Flow Control

- **Function**: Exchanges buffer credit information (e.g., PH, PD) via DLLPs between transmitter and receiver.
- **Purpose**: Prevents buffer overflow by informing TL of available space, gating TLP generation.

Data Link Layer Packets (DLLPs)

DLLPs are 8-byte packets generated by the DLL for link management, distinct from routed TLPs.

Structure



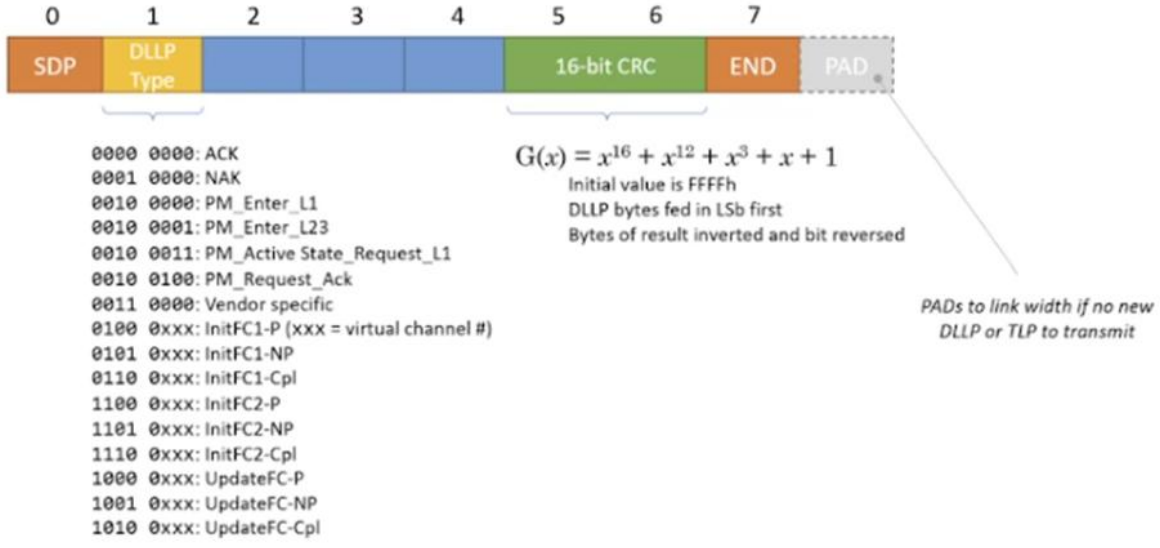
Byte 1	Byte 2	Byte 3	Byte 4
DLLP Framing Information 11110000 10101100		DLLP Type	DLLP Type Specific Information
Byte 5	**Byte 6**	**Byte 7**	**Byte 8**
DLLP Type Specific Information		LCRC	

Figure 2: Byte-wise information of 8-byte DLLP Structure in Gen3/4/5

- **Size**: 8 bytes.
- **Fields**: 2 bytes framing (start/end symbols, handled by PL). 2 bytes LCRC (16-bit CRC for DLLP integrity). 4 bytes payload (type-specific data, e.g., Sequence Number for ACK/NAK).

Characteristics

- **Non-Routed**: Used for nearest-neighbor communication only (e.g., between adjacent ports).
- **No Payload**: Unlike TLPs, DLLPs carry control info, not data.
- **Types**: ACK, NAK, InitFC (1/2), UpdateFC, Power Management, Vendor-Specific.
- **Error Handling**: Discarded if corrupted (LCRC mismatch), with periodic updates mitigating loss.



1. **ACK/NAK**: **ACK**: Confirms successful TLP receipt (e.g., Sequence Number 5 acknowledges all prior TLPs). **NAK**: Indicates TLP error (e.g., LCRC mismatch), triggers retransmission.
2. **Flow Control**: **InitFC1/InitFC2**: Initialize credits for P, NP, Cpl categories post-reset. **UpdateFC**: Updates credits during runtime.
3. **Power Management**: Controls device power states.

Behavior

- Sent periodically, updating prior info (e.g., ACK for Sequence 5 implies all prior TLPs acknowledged).
- No acknowledgment protocol for DLLPs; recovery relies on timeouts.

TLP & DLLP Processing in Data Link Layer

The DLL processes two packet types:

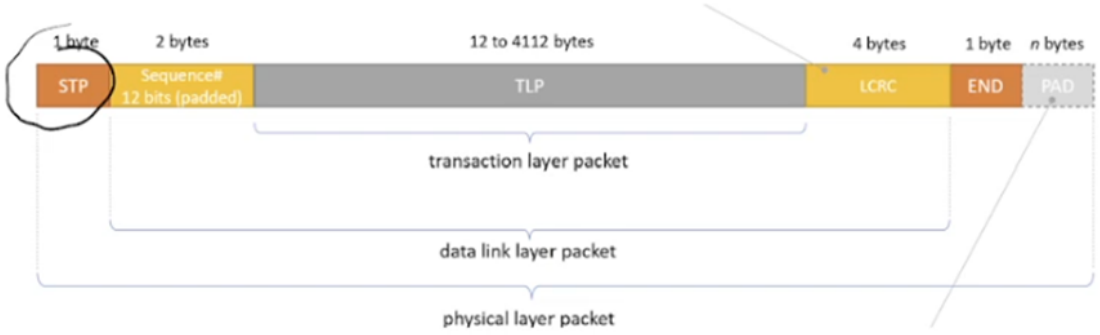
TLP Processing

- **Input**: TLPs from TL (header, payload, optional ECRC).
- **Modification**: Adds 12-bit Sequence Number (header) and 32-bit LCRC (footer).
- **Output**: Framed TLPs to PL for transmission.
- **Purpose**: Ensures integrity and order across routed paths.

DLLP Processing

- **Creation**: Generated by DLL (e.g., ACK/NAK based on TLP receipt).
- **Structure**: 4-byte payload + 2-byte LCRC + 2-byte framing.
- **Purpose**: Local link control (non-routed).

Structure of TLP (Output of DLL)



- **Format**: **Sequence Number**: 12 bits (0-4095), tracks TLP order. **TLP Body**: From TL (header, payload, ECRC if enabled). **LCRC**: 32 bits, calculated over TLP (excluding framing symbols).

DLCMSM States

The **Data Link Control and Management State Machine (DLCMSM)** configures and monitors the link:

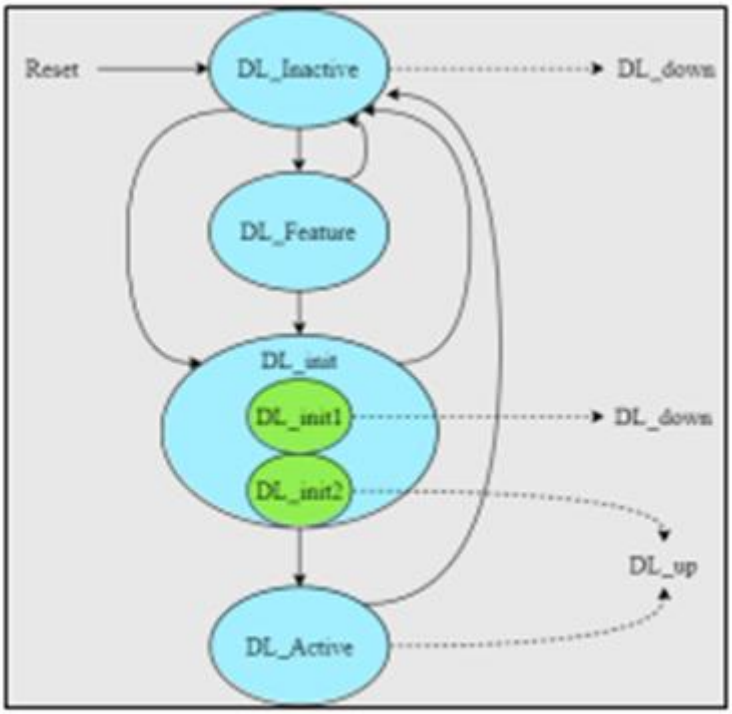


Figure 10: The DLCMSM

1. **DL_Inactive**: **Condition**: Link non-operational (e.g., disconnected or PL establishing link). **Output**: DL_Down (link paused).
2. **DL_Feature (Optional)**: **Condition**: Link operational at 16+ GT/s (e.g., Gen 5). **Action**: Exchanges feature info (e.g., buffer size) between ports.
3. **DL_Init**: **Sub-States**: **InitFC1**: Sends InitFC1 DLLPs for VC0 credits (PH, PD, NPH, NPD, CplH, CplD). **InitFC2**: Confirms credit exchange completion. **Output**: DL_Down (InitFC1), DL_Up (InitFC2). **TLPs Blocked**: Until initialization completes.
4. **DL_Active**: **Condition**: Normal operation, TLP transmission enabled. **Output**: DL_Up (link active).

Flow Control in DLL

Mechanism

- **Credit-Based**: Transmitter checks receiver buffer space via DLLPs.
- **Credit Types**: PH, PD, NPH, NPD, CplH, CplD (per VC).
- **Formula**: (Credit Limit - Cumulative Credit Required) mod 2^[Field Size] ≤ 2^[Field Size]/2 **Field Size**: 8 (headers), 12 (data). **Credit Limit (CL)**: Total credits advertised.
- **Cumulative Credit Required**: Consumed + pending credits.

Example

- **Buffer**: 1 KB (51 credits, 20 bytes each, CL = 33h).
- **Case 1**: Consumed = 20h, Pending = 01h: (33h - 21h) mod 256 ≤ 128 → True, TLP sent.
- **Case 2**: Consumed = 33h, Pending = 01h (buffer full): (33h - 34h) mod 256 ≤ 128 → False (255 > 128), TLP halted.
- **Case 3**: UpdateFC adds 2 credits (CL = 35h): (35h - 34h) mod 256 ≤ 128 → True, TLP resumes.

Scaled Flow Control

- **Need**: Higher speeds (32 GT/s) require larger buffers beyond 127 headers/2047 data credits.
- **Solution**: Scales credits (e.g., ×4 or ×16) using a scale factor in DLLPs.
- **Example**: 3-bit field (7 max) scaled by 4 = 28 (11100b).

Retry Mechanism in DLL

Purpose

- Ensures data integrity by retransmitting corrupted/lost TLPs.

Triggers

1. **NAK**: Receiver detects error (e.g., LCRC mismatch).
2. **REPLAY_TIMER Expiry**: No ACK/NAK within timeout.

Components

- **Retry Buffer**: Stores TLPs until acknowledged.
- **REPLAY_TIMER**: Max wait time for ACK/NAK, resets on ACK/NAK or TLP send.
- **ACKNAK_LATENCY_TIMER**: Delays ACK to batch TLPs, resets on ACK/NAK send.
- **REPLAY_NUM**: 2-bit counter (0-3), triggers retraining on rollover (11→00).

- **NEXT_RCV_SEQ**: Expected Sequence Number at receiver.
- **ACKD_SEQ**: Last acknowledged Sequence Number.
- **NAK_SCHEDULER**: Flag for pending NAK, blocks further ACK/NAK.
- **ACKNAK_SEQ_NUM**: Last correctly received TLP Sequence Number.

Examples

Normal Operation (ACK Case)

1. **Device A**: Sends TLPs (e.g., Sequence Numbers 6-10).

2. **Device B**:

- Receives TLP 6 successfully, schedules ACK (ACKNAK_LATENCY_TIMER starts).
- Receives TLPs 7-8 before timer expires, updates expected Sequence Number.
- Timer expires → Sends ACK with Sequence Number 8.

3. **Device A**:

- Receives ACK 8, purges TLPs ≤ 8 from retry buffer.
- Resets REPLAY_TIMER (forward progress made).
- Timer restarts if buffer has TLPs, else stays at 0.

Error Case (NAK Case)

1. **Device A**: Sends TLPs 4095, 0-3.
2. **Device B**: Receives TLP 4095, then detects CRC error in TLP 0. Sends NAK with Sequence Number 4095 immediately (no latency wait), resets ACKNAK_LATENCY_TIMER. Drops subsequent TLPs (1-3) as out-of-order.
3. **Device A**: Receives NAK 4095, purges ≤ 4095, resends 0-3. Resets REPLAY_TIMER, starts counting post-retransmission.

NOTE: REPLAY_NUM is a 2-bit counter, it will count 00,01,10,11 and then rolls back to 00 when it rolls back, a fatal error is generated and the link will go into the inactive state and link retraining should be initiated.

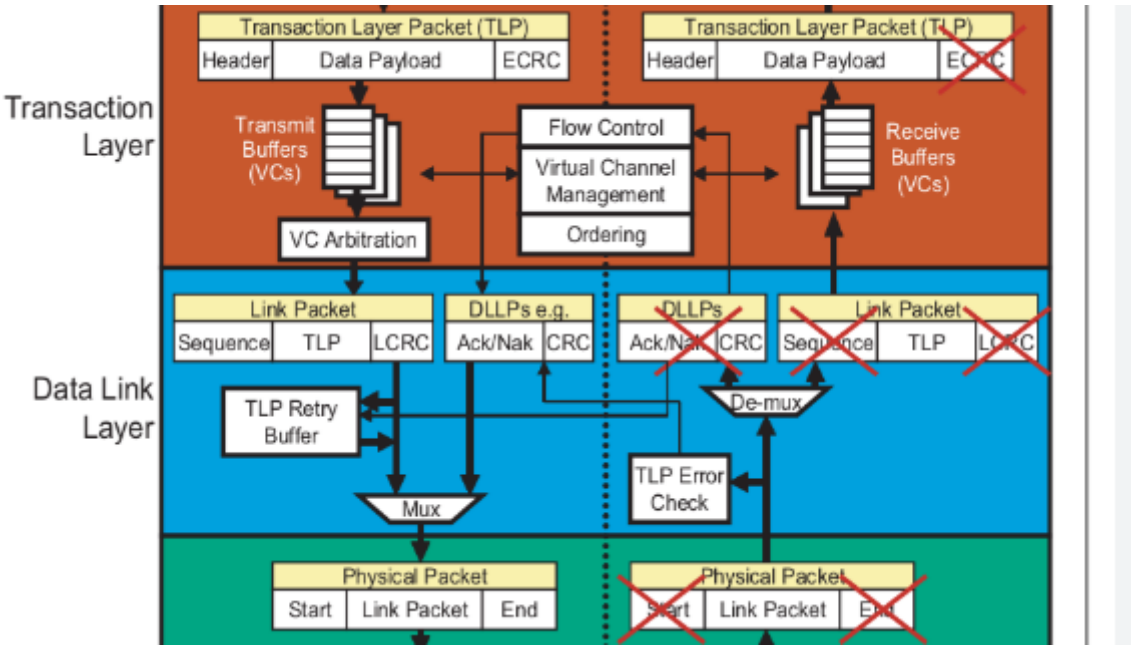
Timeout Case (RETRY_TIMER Expiry)

Receiver has Sent ACK for TLPs till sequence number 15 and has set Next_SEQ_NUM as 16. TLP with sequence number 16 is lost during transmission and the receiver receives TLP with sequence number 17.

Since Next_SEQ_NUM is not the same as the received TLP sequence number, the receiver schedules a NAK with seq number 15 and rejects the TLP with the sequence number 17.

This allows the transmitter to understand that TLPs till 15 were received correctly and TLP with sequence number 16 was not exchanged. Transmitter sends a TLP with sequence number 16 and data exchange continues without problems.

Test Plan for PCIe Data Link Layer (DLL)



Introduction

The PCIe Gen 5 Data Link Layer (DLL) facilitates reliable communication between the Transaction Layer (TL) and Physical Layer (PL). Its key responsibilities include:

- **Data Exchange**: Processing Transaction Layer Packets (TLPs) and Data Link Layer Packets (DLLPs).
- **Error Detection**: Using Link CRC (LCRC) to ensure TLP integrity.
- **Flow Control**: Managing buffer credits to prevent overflow.
- **Retry Mechanism**: Ensuring data integrity through retransmission.
- **Link Management**: Supervised by the Data Link Control and Management State Machine (DLCMSM).

This test plan verifies these functionalities through targeted test cases, covering normal operation, error conditions, and edge cases.

2. Test Categories

The test plan is divided into the following categories based on the DLL's responsibilities:

- **DLCMSM States and Transitions**
- **TLP Processing**
- **DLLP Processing**
- **Flow Control Mechanism**

- **Retry Mechanism**
- **Error Handling**
- **Performance**

3. Test Cases

3.1 DLCMSM States and Transitions

The DLCMSM manages link states and transitions. These tests verify its behavior under various conditions.

Test Case ID	Description	Objective	Steps	Expected Outcome
1.1	DL_Inactive State	Verify DLL starts in DL_Inactive when link is down	1. Simulate a non-operational link (e.g., PL link is down). 2. Check DLCMSM state and output.	DLCMSM is in DL_Inactive, output is DL_Down.
1.2	DL_Feature State	Verify transition to DL_Feature at Gen 5 speeds	1. Transition from DL_Inactive to DL_feature. 2. Monitor state and feature exchange.	DLCMSM enters DL_Feature, feature info exchanged. output is DL_Down.
1.3	DL_Init State	Validate flow control initialization	1. Transition from DL_feature to DL_Init. 2. Observe InitFC1/InitFC2 DLLPs. 3. Check state progression.	InitFC1 sent, then InitFC2, transitions to DL_Active with DL_Up.
1.4	TLP Blocking in DL_Init	Ensure TLPs are blocked until initialization completes	1. Send TLPs during DL_Init. 2. Check if TLPs are transmitted.	TLPs are blocked until DL_Active is reached.

3.2 TLP Processing

The DLL processes TLPs by adding Sequence Numbers and LCRC for transmission and validating them upon reception.

Test Case ID	Description	Objective	Steps	Expected Outcome
2.1	Outgoing TLP Modification	Verify Sequence Number and LCRC addition	1. Send TLP from TL to DLL. 2. Capture TLP sent to PL.	TLP has valid 12-bit Sequence Number and 32-bit LCRC.
2.2	Incoming TLP Validation	Confirm valid TLPs are passed to TL	1. Send TLP with correct LCRC and Sequence Number. 2. Check TL receipt.	TLP is passed to TL successfully.
2.3	Invalid TLP Handling	Ensure invalid TLPs trigger error handling	1. Send TLP with incorrect LCRC or Sequence Number. 2. Observe response.	TLP discarded, NAK sent, or retry triggered.

3.3 DLLP Processing

DLLPs handle link management tasks such as ACK/NAK, flow control, and power management.

Test Case ID	Description	Objective	Steps	Expected Outcome
3.1	ACK DLLP Generation	Verify ACK for successful TLPs	1. Send valid TLPs. 2. Monitor DLLP response.	ACK DLLP sent with correct Sequence Number.
3.2	NAK DLLP Generation	Test NAK for errored TLPs	1. Send TLP with invalid LCRC. 2. Check DLLP response.	NAK DLLP sent with last valid Sequence Number.
3.3	InitFC DLLP Exchange	Confirm flow control initialization DLLPs	1. Monitor DLLPs during DL_Init state.	InitFC1 and InitFC2 DLLPs sent for all credit types.
3.4	UpdateFC DLLP Transmission	Ensure periodic credit updates	1. Operate link in DL_Active. 2. Monitor UpdateFC DLLPs.	UpdateFC DLLPs sent periodically.
3.5	Power Management DLLP	Validate power state control	1. Send Power Management DLLPs. 2. Check device state transitions.	Power states updated without disrupting link.

3.5 Retry Mechanism

The retry mechanism ensures data integrity by retransmitting lost or corrupted TLPs.

Test Case ID	Description	Objective	Steps	Expected Outcome
5.1	NAK-Triggered Retry	Verify retransmission after NAK	1. Send TLP with LCRC error. 2. Check for NAK and retransmission.	NAK sent, TLP retransmitted successfully.
5.2	REPLAY_TIMER Expiry	Test retransmission on timeout	1. Block ACK/NAK response. 2. Wait for REPLAY_TIMER expiry.	TLPs retransmitted after timeout.
5.3	REPLAY_NUM Rollover	Ensure link retraining on excessive retries	1. Force multiple retransmissions until REPLAY_NUM rolls over.	Link enters DL_Inactive, retraining initiated.

3.6 Error Handling

Error handling tests verify the DLL's response to various fault conditions.

Test Case ID	Description	Objective	Steps	Expected Outcome
6.1	LCRC Error Detection	Confirm LCRC mismatch detection	1. Send TLP with incorrect LCRC. 2. Check response.	Error detected, NAK sent or retry triggered.
6.2	Sequence Number Error	Test handling of out-of-order TLPs	1. Send TLPs with incorrect Sequence Numbers. 2. Observe behavior.	Out-of-order TLPs discarded, NAK sent.
6.3	Flow Control Violation	Verify response to insufficient credits	1. Send TLP with no credits available. 2. Check system behavior.	Transmission prevented, no data loss occurs.

3.7 Performance and Timing

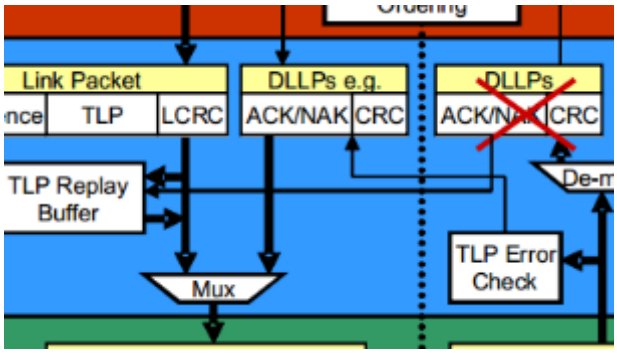
These tests ensure the DLL meets timing requirements at Gen 5 speeds.

Test Case ID	Description	Objective	Steps	Expected Outcome
7.1	Timing Compliance	Verify operations within PCIe Gen 5 timing specs	1. Measure TLP/DLLP processing latencies at 32 GT/s.	All timings within specification limits.
7.2	High-Speed Operation	Test DLL at maximum speed	1. Run link at 32 GT/s with continuous TLPs. 2. Monitor for errors.	No timing violations or data corruption.

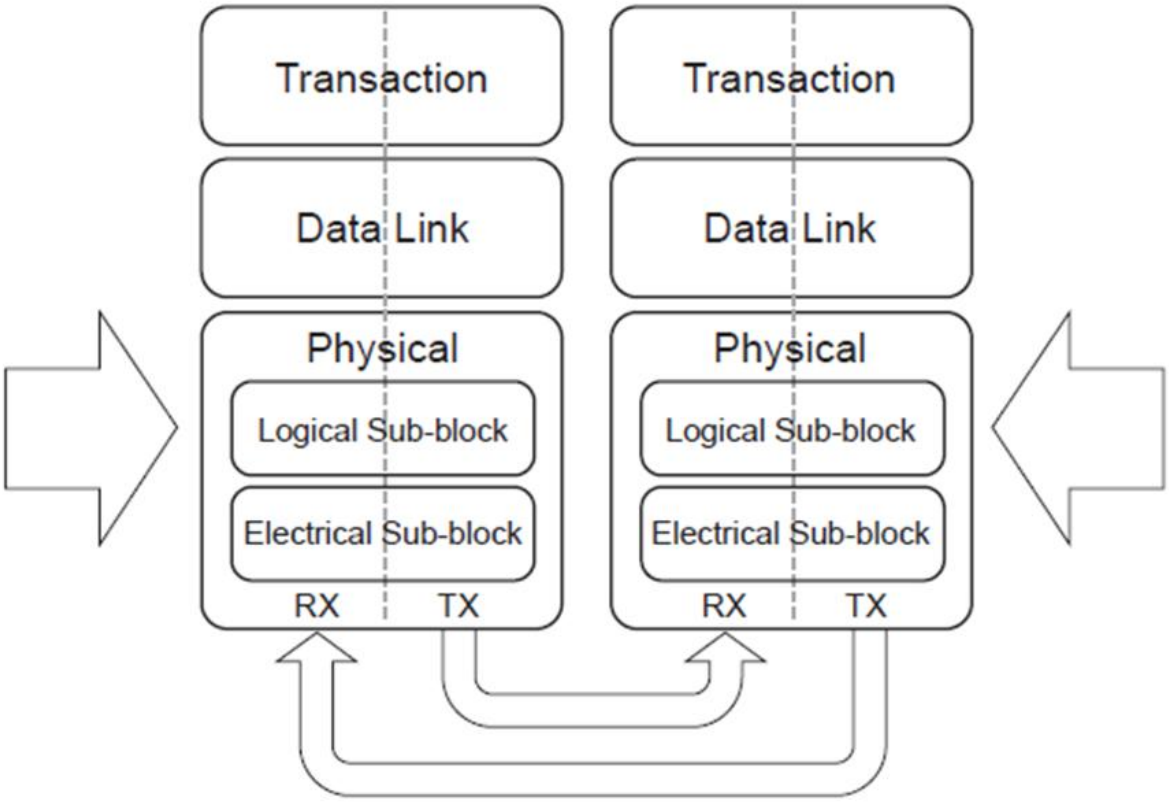
4. Additional Considerations

- **Scalability**: Test different link widths (e.g., x1, x16) and speeds (16 GT/s, 32 GT/s).
- **Coverage**: Define functional coverage points for states, transitions, and error scenarios.
- **Tools**: UVM testbenches with PCIe Gen 5 Verification IP (VIP).

PCIe Physical Layer part - 1: Overview



The PCIe Physical Layer (PHY) is responsible for the reliable transmission and reception of data across the physical medium connecting two PCIe devices. It interfaces with the Data Link Layer (DLL) above it and manages the electrical signaling through the Electrical Sub-block. The Logical Sub-block, a key component of the PHY, handles data preparation, encoding, framing, and coordination with the Electrical Sub-block to ensure proper link operation.



Logical Sub-block: Structure and Functionality

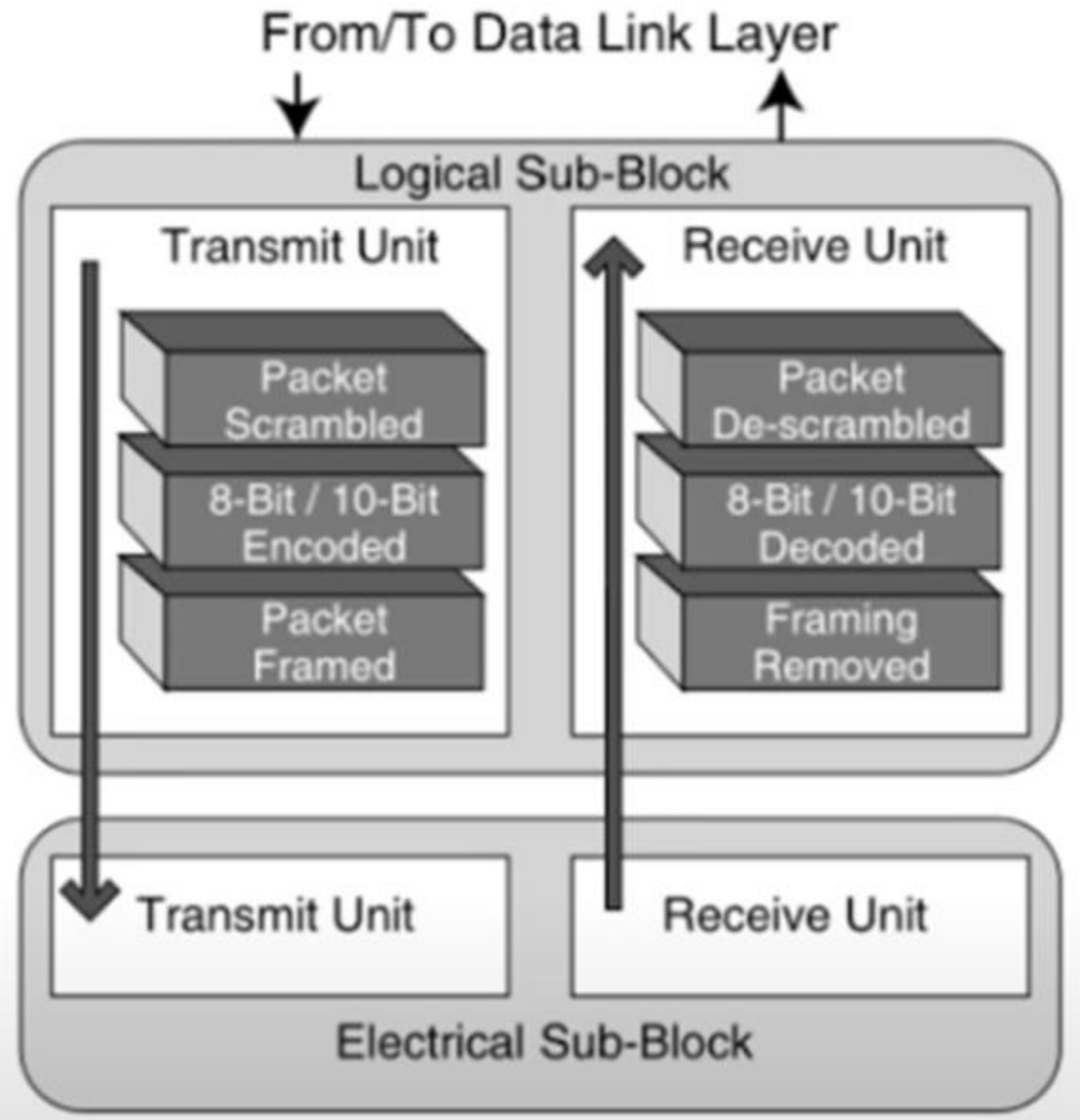
The Logical Sub-block within the PCIe Physical Layer is divided into two primary sections:

1. **Transmit Section**: **Purpose**: Prepares outgoing data received from the Data Link Layer for transmission. **Process**: Encodes, frames, and scrambles the data (as needed) before passing it to the Electrical Sub-block for transmission over the physical medium.
2. **Receiver Section**: **Purpose**: Processes incoming data from the Electrical Sub-block before forwarding it to the Data Link Layer. **Process**: Identifies received symbols, decodes them, checks for errors, and removes framing before delivering the data.

Electrical Sub-block:

The **electrical sub-block** of the PCIe PHY layer handles analog signal transmission and reception. Key functions:

- **Differential signaling** (noise-resistant TX/RX pairs).
- **Transmitters** (convert digital to analog) and **receivers** (convert analog to digital).
- **Impedance matching** (100Ω termination) and **signal integrity** (equalization, EMI reduction).
- **Clock recovery** (CDR synchronizes data sampling).
- Complies with PCIe **voltage/jitter standards** and supports **low-power states**.
- **SerDes** (serialize/deserialize data) and **calibration** (self-test for signal quality).



Encoding Schemes in the Logical Sub-block

The Logical Sub-block employs different encoding schemes depending on the PCIe data rate, as specified in the document:

1. 8b/10b Encoding (2.5 GT/s and 5.0 GT/s)

- **Description**: Used for PCIe data rates of 2.5 GT/s and 5.0 GT/s, this scheme maps 8-bit data characters into 10-bit symbols.
- **Mechanism**: Each 8-bit character is split into a 3-bit group (mapped to a 4-bit code) and a 5-bit group (mapped to a 6-bit code), forming a 10-bit symbol. A control bit identifies when to encode one of 12 **Special Symbols** (e.g., COM, STP, END) for link management and framing.
- **Purpose**: Ensures DC balance (equal numbers of 0s and 1s over time) and to embed the clock signal in the data stream.

Special Symbols for Framing and Link Management

Examples:

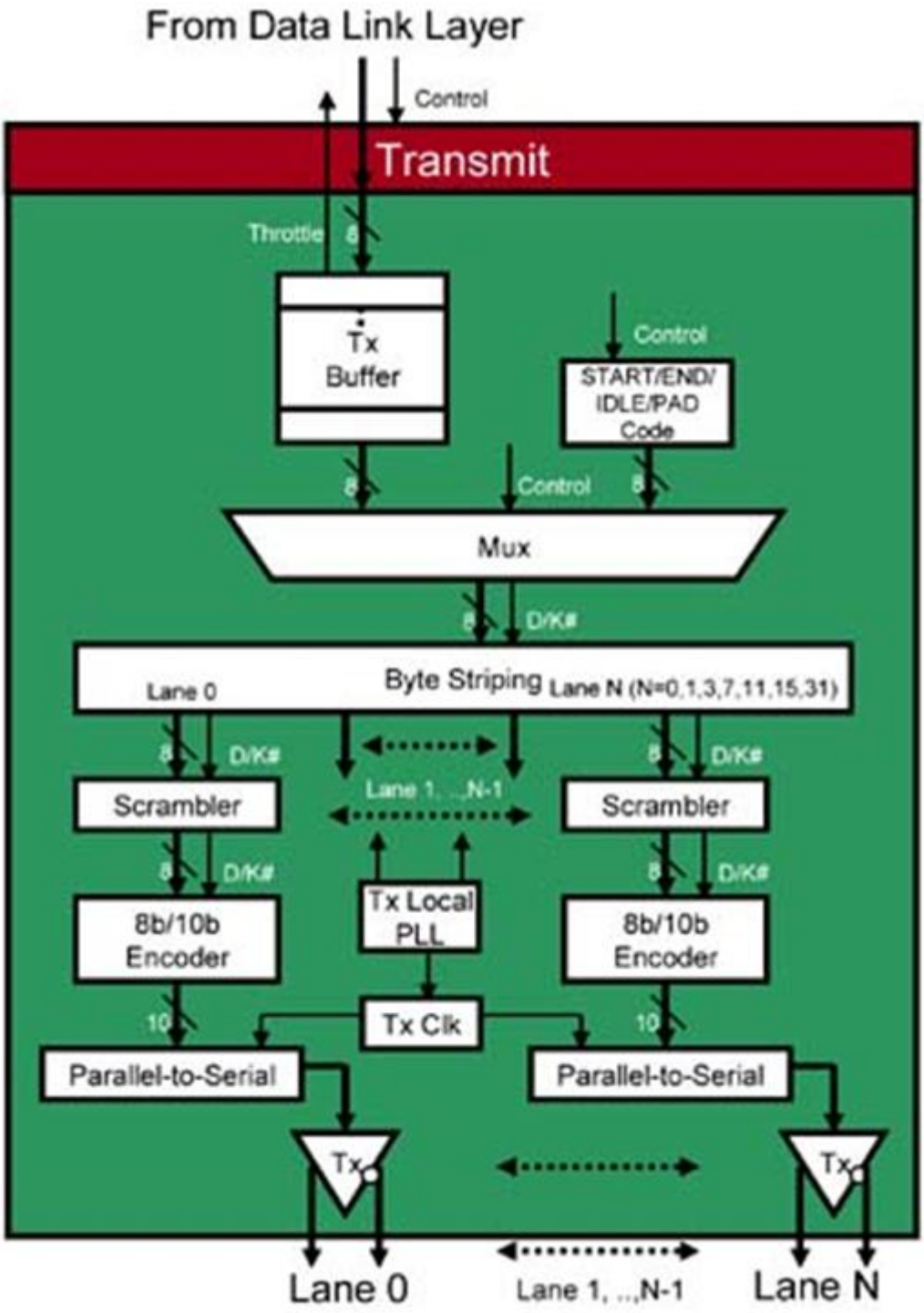
- **COM (K28.5)**: Initializes lanes and links.
- **STP (K27.7)**: Marks the start of a Transaction Layer Packet (TLP).
- **SDP (K28.2)**: Marks the start of a Data Link Layer Packet (DLLP).
- **END (K29.7)**: Marks the end of a TLP or DLLP.
- **SKP (K28.0)**: Compensates for clock differences between ports.
- **Usage**: These symbols frame TLPs and DLLPs and support link operations like idle states and training.

Framing Rules

- **TLPs**: Framed with an STP symbol at the start and an END (or EDB for nullified TLPs) at the end.
- **DLLPs**: Framed with an SDP symbol at the start and an END symbol at the end.
- **Ordered Sets**: for Link Training.
- **Logical Idle**: When no packets are transmitted, the link sends scrambled idle data (0x00) to maintain synchronization.

Scrambling

- **Purpose**: Reduces electromagnetic interference (EMI) by randomizing data patterns.
- **Method**: Uses a Linear Feedback Shift Register (LFSR) with the polynomial G(X) = X^16 + X^5 + X^4 + X^3 + 1.
- **Rules**: Data symbols (D codes) are scrambled, but special symbols (K codes) are not. The LFSR is initialized with a seed of FFFFh after a COM symbol and advances for each data symbol except SKP.



2. 128b/130b Encoding (8.0 GT/s and Higher)

- **Description**: Used for data rates of 8.0 GT/s and above, this scheme employs a per-lane block code with a 2-bit Sync Header and a 128-bit payload.
- **Mechanism**: **Sync Header**: 10b: Indicates a Data Block (128-bit payload containing TLPs, DLLPs, or idle data). 01b: Indicates an Ordered Set Block (e.g., SKP, EIEOS for link management). Blocks are 130 bits total (2-bit header + 128-bit payload).

- **Purpose**: Improves efficiency at higher data rates while maintaining synchronization and error detection.

Framing Tokens

- **Examples**: **STP**: Start of a TLP (4 symbols, includes sequence number). **SDP**: Start of a DLLP (2 symbols). **EDB**: End of a nullified TLP (4 symbols). **EDS**: End of Data Stream (4 symbols, precedes an Ordered Set). **IDL**: Logical Idle (1 symbol, sent when no packets are transmitted).
- **Usage**: These tokens define the boundaries of TLPs and DLLPs within Data Blocks.

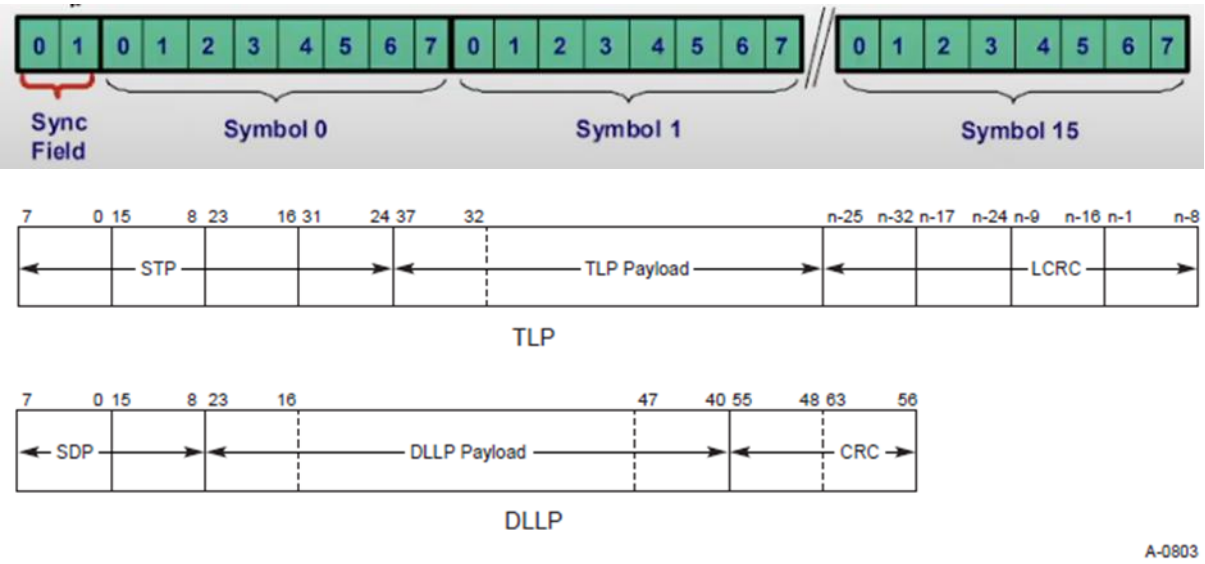


Figure 4-14 TLP and DLLP Layout

Scrambling

- **Method**: Uses an LFSR with the polynomial G(X) = X^23 + X^21 + X^16 + X^8 + X^5 + X^2 + 1.
- **Rules**: Sync Headers and certain Ordered Sets (e.g., SKP, EIEOS) bypass scrambling. Data Block payloads are scrambled to reduce EMI. Each lane may have its own LFSR, initialized with a lane-specific seed.

Precoding (32.0 GT/s and Higher)

- **Purpose**: Enhances signal integrity at very high data rates.
- **Mechanism**: Applied to scrambled bits, requested by the receiver during link training via EQ TS2 Ordered Sets.
- **Rules**: Only scrambled bits are precoded, with the "previous bit" reset to 1b at block boundaries.

Multi-Lane Link Operation

- **Symbol Striping**: In multi-lane links (e.g., x4, x8), symbols are distributed across lanes. For example, in a x4 link, byte 0 goes to Lane 0, byte 1 to Lane 1, etc.
- **Framing Placement**: STP and SDP symbols must start on Lane 0 (or Lane 4*N for wider links) when transitioning from Logical Idle. END or EDB symbols may be followed by PAD or IDL symbols to fill unused lanes in wider links.

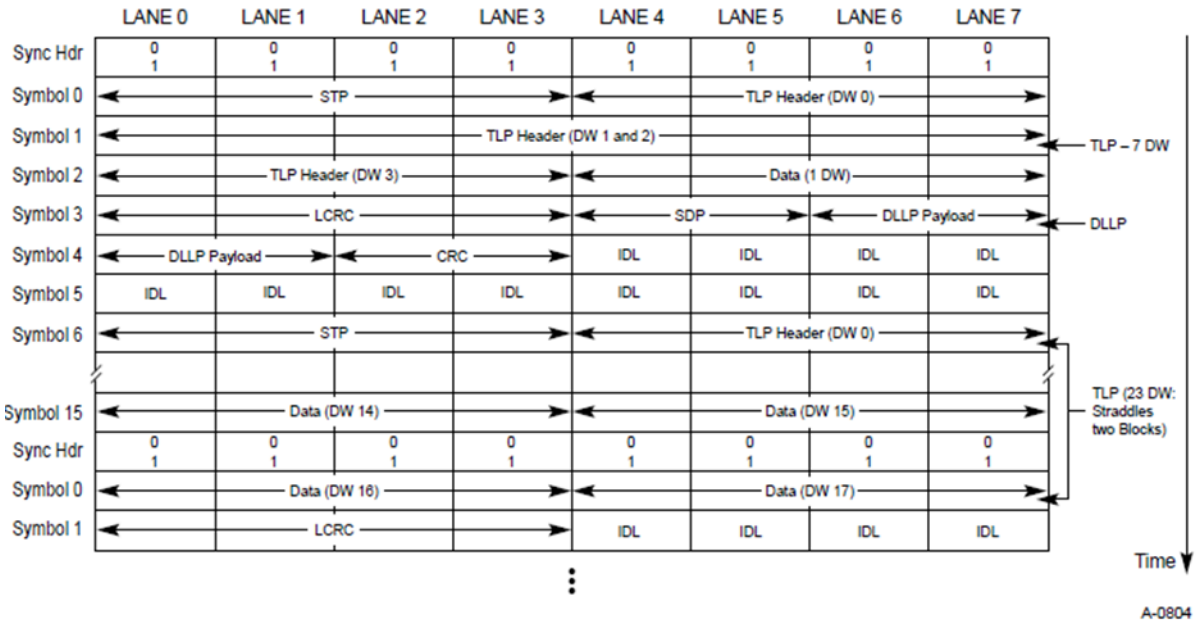
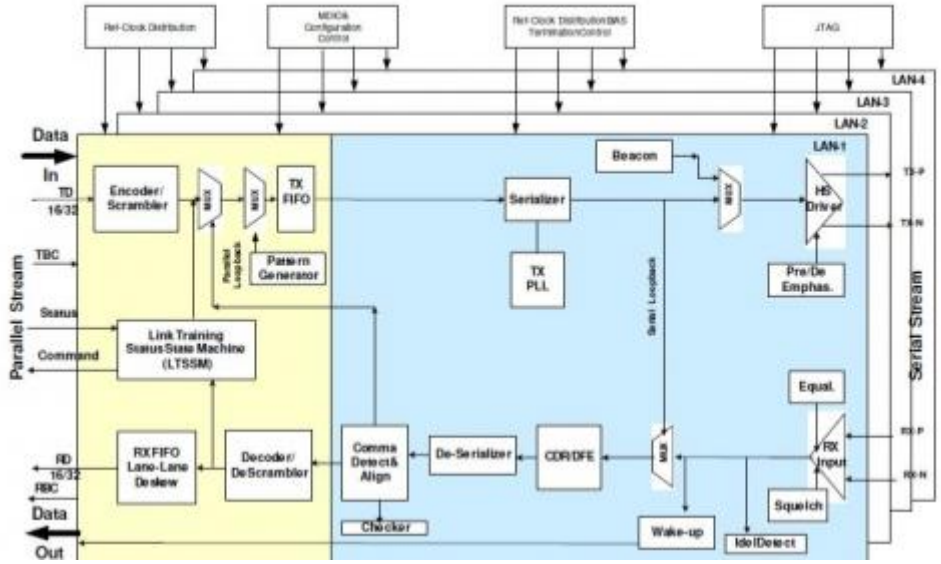


Figure 4-15 Packet Transmission in a x8 Link

Link Management Features

1. **Loopback Mode**: **Purpose**: Facilitates testing and debugging. **Operation**: The receiver (slave) retransmits received bits without modification, except for SKP Ordered Sets, which may be adjusted for clock compensation.
2. **Link Equalization (8.0 GT/s and Higher)**: **Purpose**: Optimizes transmitter and receiver settings for high-speed operation. **Process**: Involves multiple phases during link training, initiated autonomously or by software, to ensure signal quality.

PCIE Physical Layer Part – 2: Link Training and Status State Machine (LTSSM)

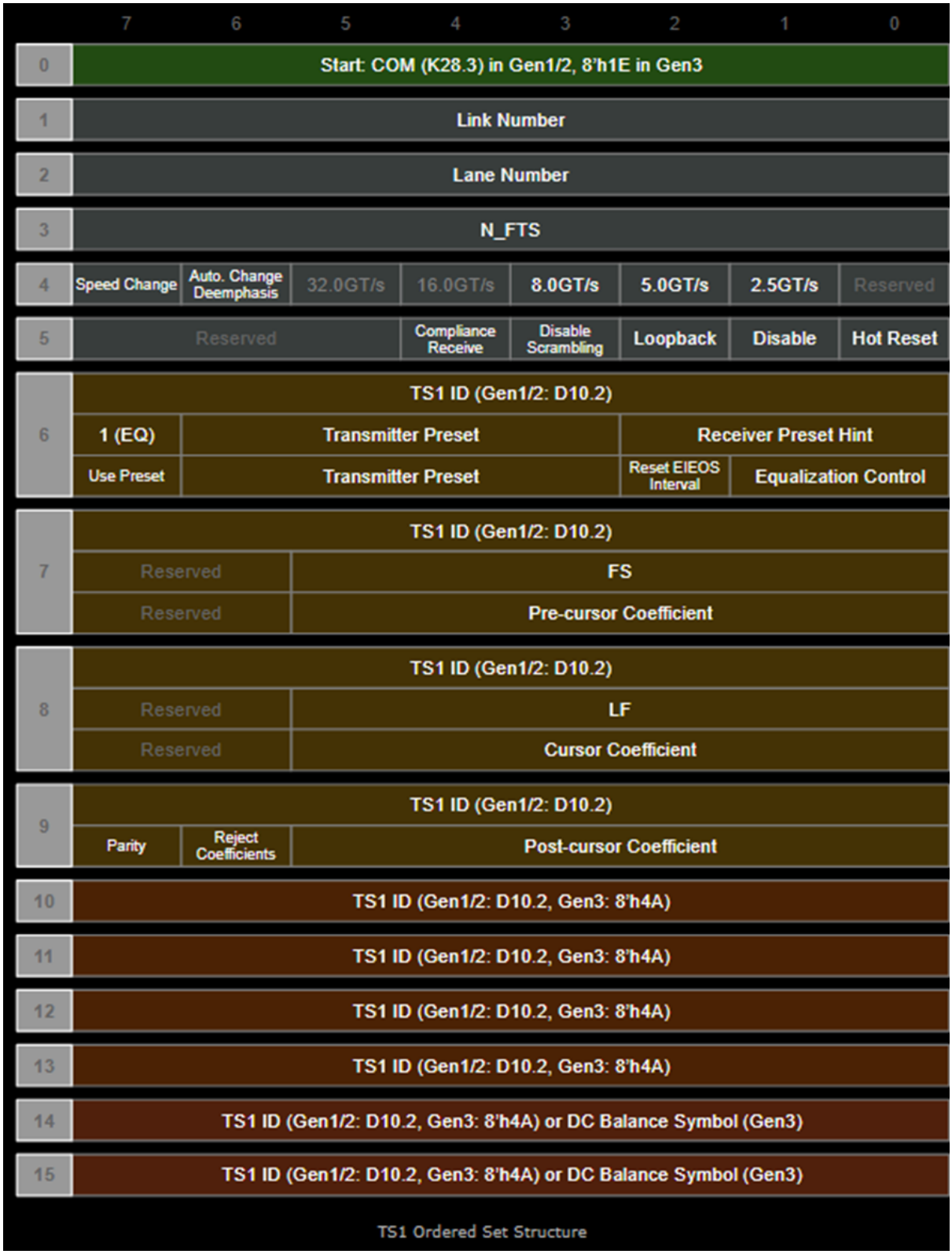


The **Link Training and Status State Machine (LTSSM)** is a finite state machine within the PCIe physical layer that ensures a link between two devices (e.g., a Root complex and an endpoint) is properly established and maintained. It manages the link's initialization, configuration, operation, error recovery, and power states. The LTSSM operates through a series of defined states, transitioning between them based on conditions like received signals, timeouts, or directives from higher layers.

Key Processes in the LTSSM

The LTSSM oversees several processes critical to PCIe link functionality:

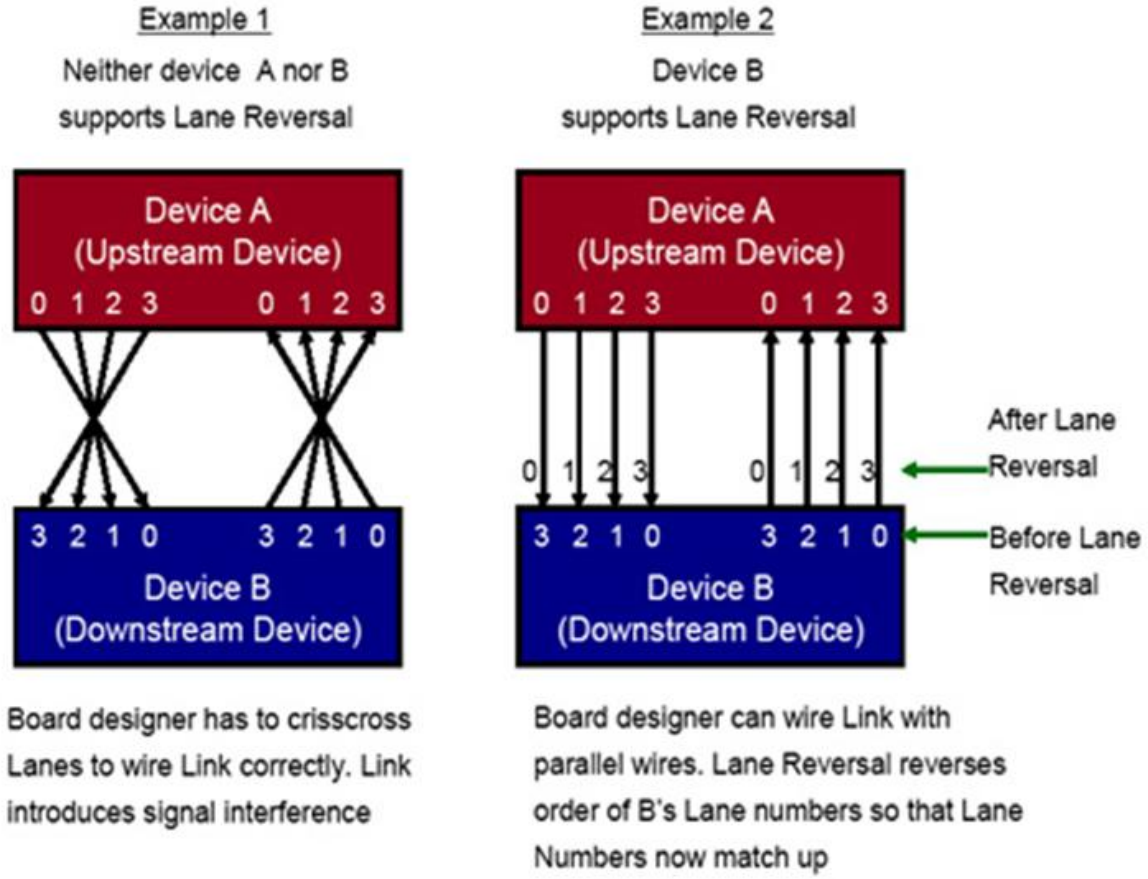
- **Sequence**: Begins in the *Detect* state, progresses to *Polling* for alignment, then *Configuration* for parameter negotiation, and finally reaches *L0* for normal operation.
- **Training Sequences**: Uses **TS1** and **TS2 Ordered Sets** to exchange physical layer parameters like link width, lane numbers, and data rates.

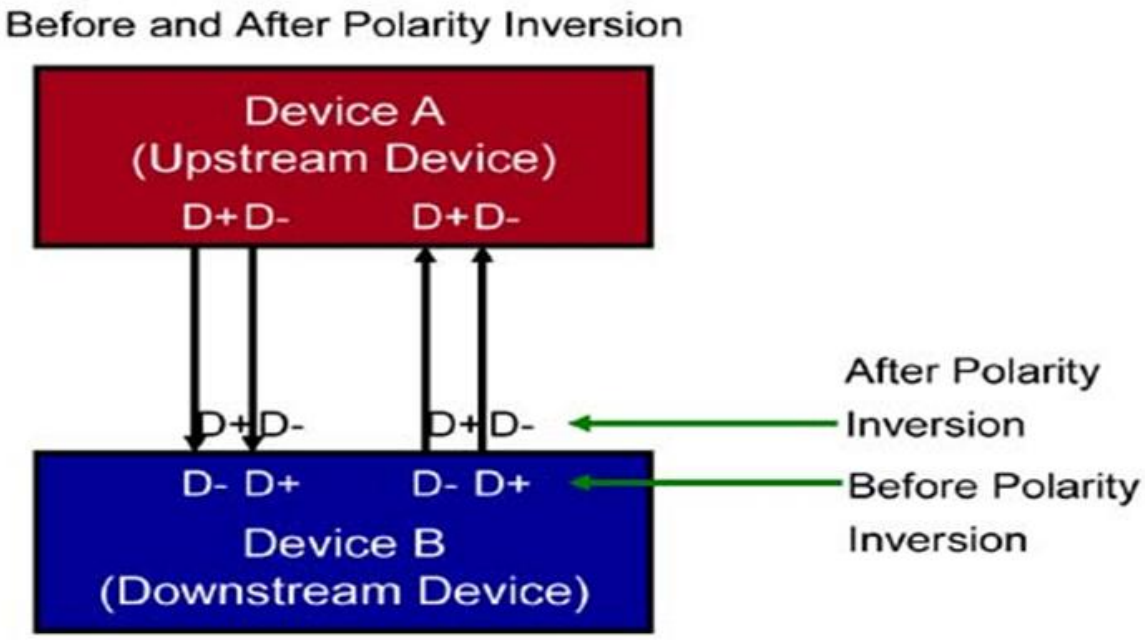


TS1 Ordered Set Structure

TS2 Ordered Set Structure

- **Data Rate Negotiation** Devices advertise supported data rates (e.g., 2.5 GT/s, 5.0 GT/s, 8.0 GT/s, 16.0 GT/s, 32.0 GT/s) in TS1/TS2 Ordered Sets. The link operates at the highest mutually supported data rate.
- **Link Width Negotiation** Determines the number of active lanes (e.g., x1, x4, x8, x16) based on hardware capabilities and negotiation during the *Configuration* state.
- **Lane Reversal/Polarity Inversion**

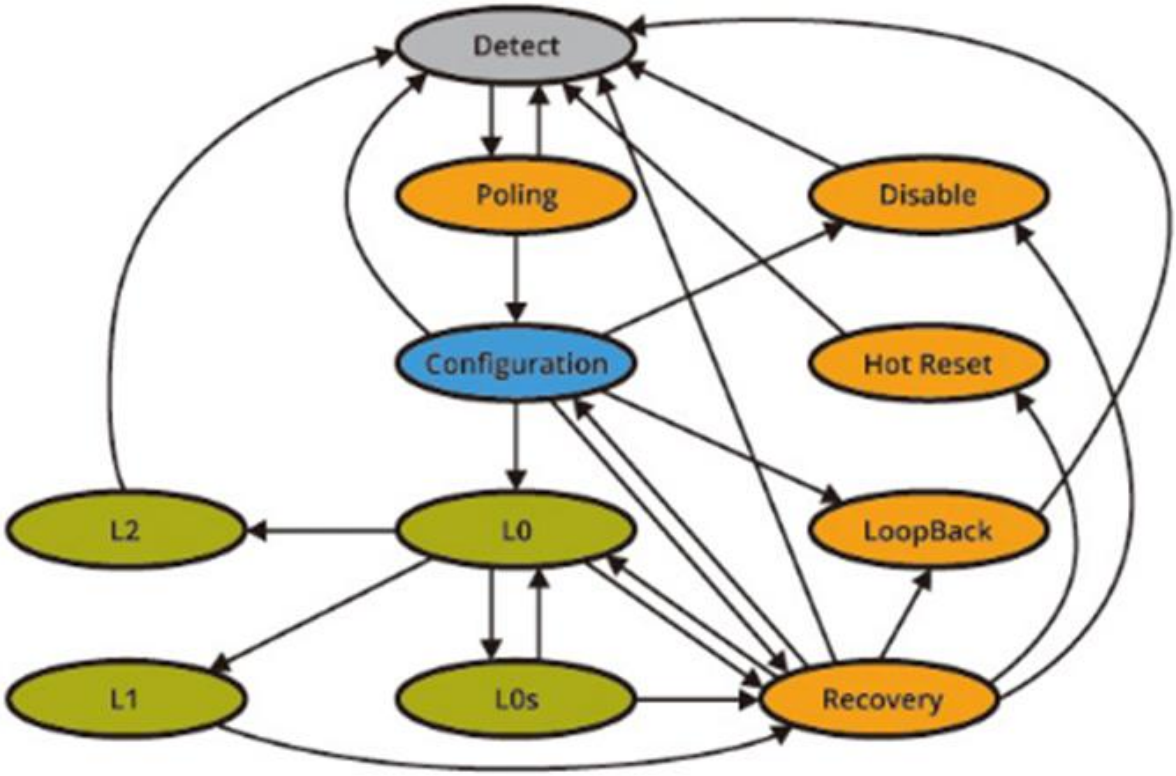




- **Bit and Symbol Lock** Ensures the receiver aligns its bit and symbol timing with the transmitter, using training sequences like TS1/TS2 and Fast Training Sequences (FTS).
- **Error Recovery** Triggered by issues like loss of symbol lock or block alignment, moving the link to the *Recovery* state to retrain.

- **Power Management** Supports low-power states (*L0s*, *L1*, *L2*) to reduce power consumption during idle periods.

LTSSM States and Substates



1. **Detect**: Checks for the presence of a connected device by detecting electrical signals. Its purpose is to periodically look for receiver termination, indicating the presence of a link partner.
2. **Polling**: Exchanges training sequences (TS1/TS2 ordered sets) to synchronize transceivers and align bit/symbol boundaries. **Purpose:** Bit Lock, Symbol Lock, Lane Polarity, Lane Data Rate, Compliance testing also occurs in this state. During compliance testing, the transmitter outputs a specified compliance pattern. This is intended to be used with test equipment to verify that all of the voltage, noise emission and timing specifications are within tolerance. **Transitions**: On success, moves to **Configuration**; on failure, returns to Detect.
3. **Configuration**: **Action**: Negotiates link parameters, Link width, Link Number, Lane reversal, Polarity inversion (if necessary), Lane-to-Lane de-skew is performed. **Lane Count**: Determines the number of active lanes (e.g., x1, x4, x16). **Transitions**: Proceeds to **L0** (active state) once configured.

4. **L0 (Active State)**: **Action**: Normal operation, transmitting/receiving data packets. **Power Management**: Can transition to low-power states (**L0s**, **L1**) during idle periods. **Error Handling**: Moves to **Recovery** if errors exceed thresholds.

5. **Recovery**: **Action**: Retrains the link to recover from errors (e.g., bit errors, clock drift), speed changes. **Process**: Re-enters **Polling** and **Configuration** to re-establish parameters. **Transitions**: Returns to **L0** after successful retraining; falls back to Detect on repeated failures.

6. **L0s (Low-Power Standby)**: This is a low power, Active Power Management state. It takes a very short time (in the order of 50ns) to transit from the L0s state back to the L0 state (because the LTSSM does not have to go through the Recovery state). This state is entered after a transmitter sends and the remote receiver receives Electrical Idle Ordered-Sets while in the L0 state. Exit from the L0s state to the L0 state involves sending and receiving FTS Ordered-Sets. When transitioning from L0s exit to L0, Lane-to-Lane de-skew must be performed, and Bit and Symbol Lock must be re-established.

7. **L1 (Lower Power State)**: **Action**: Deeper power-saving mode with longer wakeup latency. **Usage**: Triggered by software or prolonged inactivity.

8. **L2(Lower Power State):**This is the lowest power state. Most of the transmitter and receiver logic is powered down (with the exception of the receiver termination, which must be powered for the receiver to be in a low impedance state). Main power and the clock are not guaranteed, though Vaux power is available. When Beacon support is required by the associated system or form factor specification, an upstream port that supports this wakeup capability must be able to send the Beacon signal and a downstream port must be able to detect the Beacon signal. Beacon signaling or the WAKE# signal is used by a device in the D3Cold state to trigger a system wakeup event (i.e., a request for main power supply re-activation). Another power state defined by the specification is the L3 state, but this state does not relate to the LTSSM states. The L3 Link state is the full-off state where the Vaux power signal is not available. A device in L3 cannot trigger a wakeup event unless power is re-applied to the device through some other mechanism.

9. **Loopback**: **Action**: Diagnostic state where a device echoes data back to the transmitter. **Purpose**: Validates transmitter/receiver integrity without external hardware.

10. **Hot Reset**: **Action**: Resets the link without power cycling (initiated by software or a TS1/TS2 ordered set). **Effect**: Reinitializes the link, similar to a cold reset but faster.

11. **Disabled**: **Action**: Non-operational state due to critical failures or software commands. **Recovery**: Requires full reinitialization or system intervention.

Detect Substate:

Detect.Quiet: This is the entry point of the LTSSM after a reset and the reset point after many timeout or fault conditions. The LTSSM stays in this state until 12ms have elapsed or the receiver detects that any lane has exited electrical idle (phy_rxelecidle goes low). Then, it will proceed to Detect.Active.

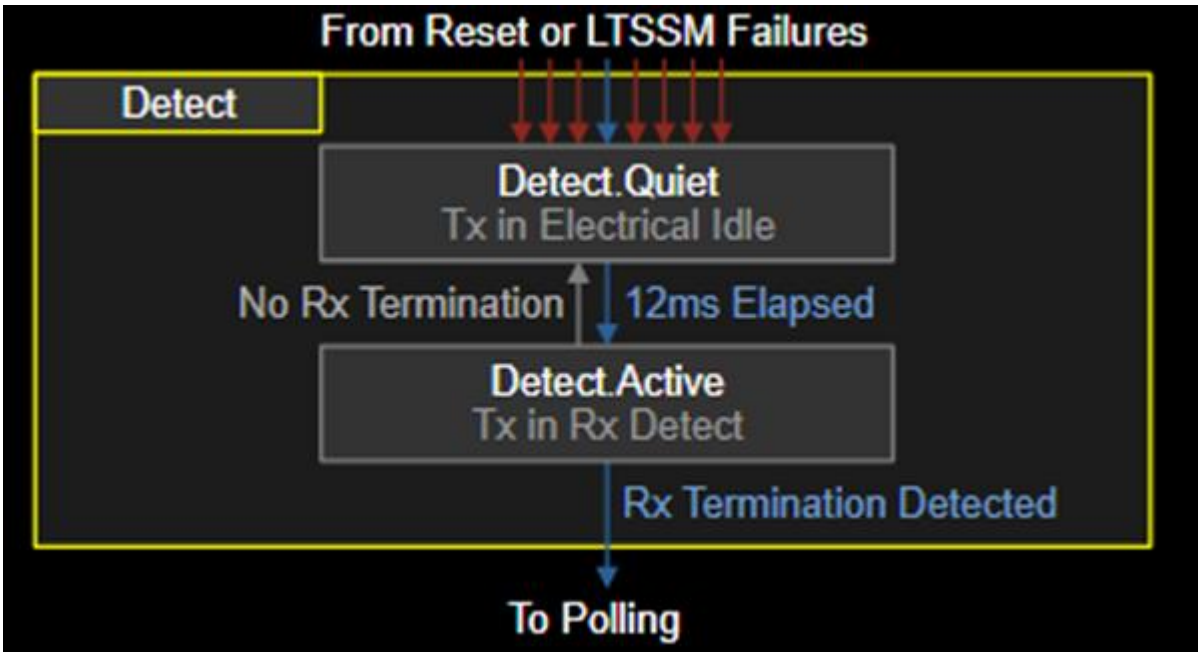
Detect.Active The transmitter for each lane attempts to detect receiver termination on that lane, indicating the presence of a link partner.

There are three possible outcomes:

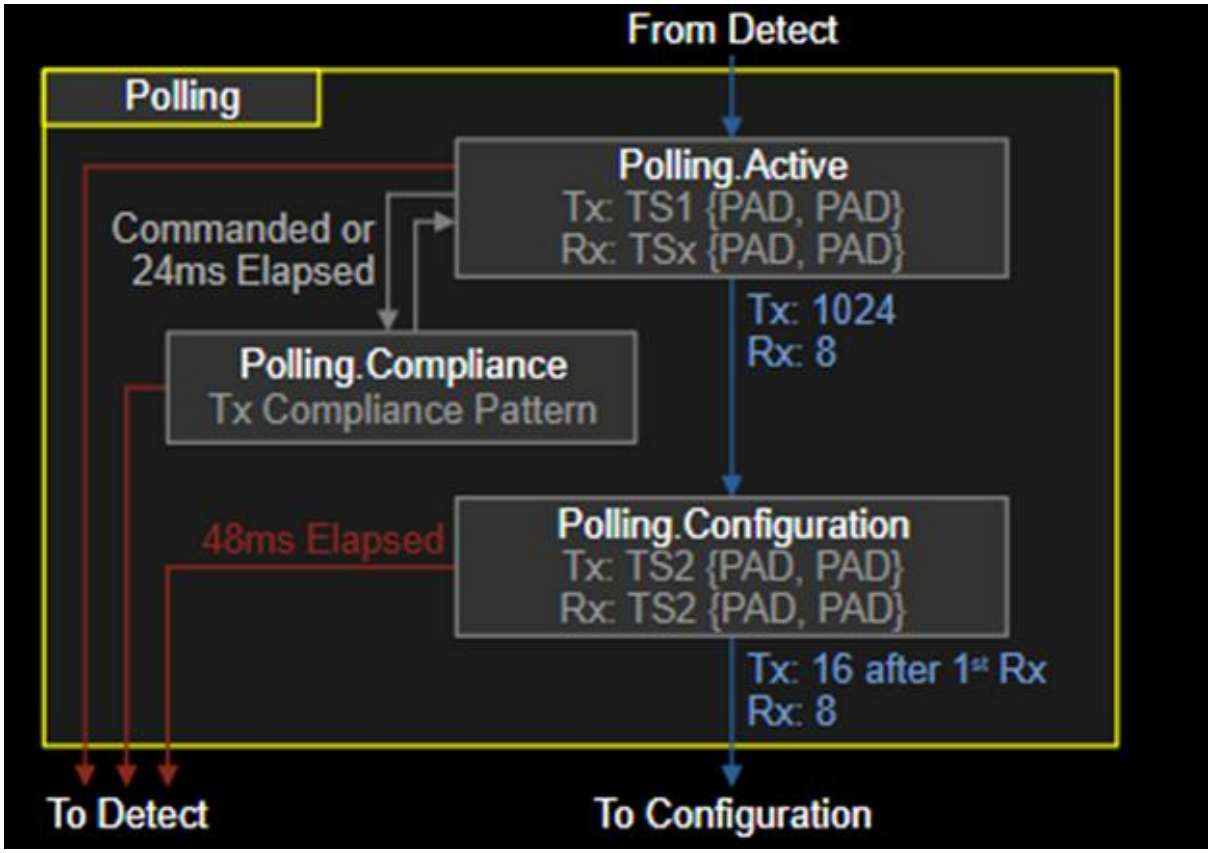
o No receiver termination is detected on any lane. The LTSSM returns to Detect.Quiet.

o Receiver termination is detected on all lanes. The LTSSM proceeds to Polling on all lanes.

o Receiver termination is detected on some, but not all, lanes. In this case, the link partner may have fewer lanes. The transmitter waits 12ms, then repeats the receiver detection. If the result is the same, the LTSSM proceeds to Polling on only the detected lanes. Otherwise, it returns to Detect.Quiet.



The Polling state consists of two substates: **Polling.Active** and **Polling.Configuration**.



Polling.Active

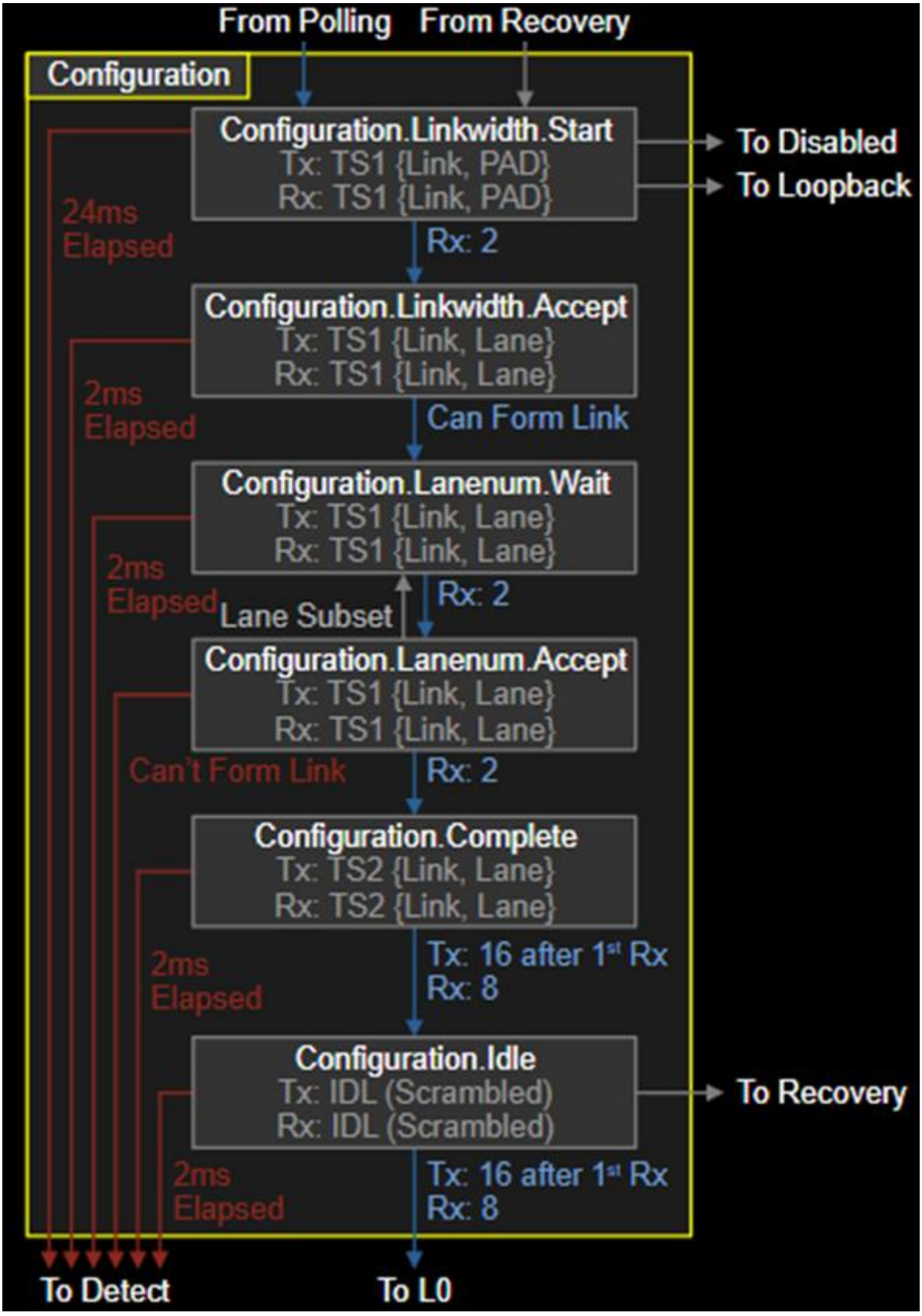
- **Transmitter Behavior**: Sends **TS1 Ordered Sets** with PAD in the Link and Lane Number fields.
- **Receiver Behavior**: Listens for TS1 or TS2 Ordered Sets from the link partner.
- **Transition to Polling.Configuration**: This occurs when the following conditions are met: At least **1024 TS1s** have been transmitted. Eight consecutive TS1s or TS2s are received with PAD for Link and Lane Numbers on **all lanes**, without requesting Polling.Compliance (unless also requesting Loopback, a rare case).
- **Timeout (24ms)**: If the above conditions are not fully met after 24ms: The LTSSM proceeds to Polling.Configuration if at least one lane meets the criteria and enough lanes have exited electrical idle to form a valid link. Otherwise, it transitions to **Polling.Compliance**, a substate for testing PCIe PHY compliance by transmitting known sequences (assuming connection to a passive test load).

Polling.Configuration

- **Transmitter Behavior**: Sends **TS2 Ordered Sets** with PAD in the Link and Lane Number fields.
- **Receiver Behavior**: Listens for TS2 Ordered Sets from the link partner (TS1s are ignored).

- **Transition to Configuration**: This occurs when: At least **16 TS2s** have been transmitted after receiving at least one TS2. Eight consecutive TS2s are received with PAD for Link and Lane Numbers on **any lane**.
- **Synchronization Mechanism**: The counting of transmitted TS2s begins only after receiving at least one TS2, ensuring both link partners are synchronized and receive sufficient TS2s to exit the state, regardless of which entered first.
- **Timeout (48ms)**: If the conditions are not met within 48ms, the LTSSM returns to the **Detect** state and restarts the process.

Configuration substates



Configuration.Linkwidth.Start

- **Purpose**: This substate initiates the link configuration process by proposing a **Link Number**—a unique identifier for the link—between the downstream and upstream ports.
- **Transmitter (Downstream Port)**: Sends **TS1 Ordered Sets** with: An arbitrary **Link Number** (chosen from 0 to 31). **PAD** for the Lane Number, indicating that lane assignments are not yet defined.
- **Receiver**: Listens for TS1 Ordered Sets from the upstream port that match the transmitted Link Number and have PAD for the Lane Number.
- **Transition to Configuration.Linkwidth.Accept**: Occurs when: **Two consecutive TS1s** are received on **any lane** with the same Link Number as transmitted and PAD for the Lane Number.
- **Timeout**: If the condition is not met within **24ms**, the LTSSM returns to the **Detect** state, restarting the link training process.

Configuration.Linkwidth.Accept

- **Purpose**: Evaluates whether a link can be formed using the lanes that responded and assigns preliminary lane numbers if possible.
- **Downstream Port**: Analyzes the lanes that sent TS1s with the matching Link Number and PAD for the Lane Number. If a link can be formed: Assigns **sequential Lane Numbers** (e.g., 0 to 3 for an x4 link) to those lanes.
- **Transition to Configuration.Lanenum.Wait**: Occurs when: A valid link can be formed with a **subset of the responding lanes**.
- **Handling Partial Responses**: The document raises a question about whether the LTSSM should wait for additional TS1s to account for missed packets or lane-to-lane skew before assigning lane numbers. While not explicitly answered, waiting a few TS periods seems logical to ensure all lanes have a chance to respond.
- **Timeout**: If no valid link can be formed within **2ms**, the LTSSM returns to **Detect**.

Configuration.Lanenum.Wait

- **Purpose**: Proposes specific lane numbers to the upstream port and waits for confirmation.
- **Transmitter (Downstream Port)**: Sends **TS1 Ordered Sets** with: The agreed **Link Number**. The proposed **Lane Numbers** for each lane (no longer PAD).
- **Receiver**: Listens for TS1s from the upstream port that echo the Link Number and the proposed Lane Numbers.

- **Transition to Configuration.Lanenum.Accept**: Occurs when: **Two consecutive TS1s** are received on **any lane** with the matching Link Number and **updated Lane Numbers** (i.e., reflecting the proposed assignments).
- **Delay Consideration**: The upstream port may take up to **1ms** to respond due to receiver errors or skew, so the transition condition is likely evaluated only after this period.
- **Timeout**: If the condition is not met within **2ms**, the LTSSM returns to **Detect**.

Configuration.Lanenum.Accept

- **Purpose**: Finalizes lane assignments by evaluating the upstream port's response and handling discrepancies or special cases like lane reversal.
- **Evaluation**: The downstream port examines the Lane Numbers received in TS1s and considers three outcomes: **Match or Reverse Match**: The received Lane Numbers match the transmitted ones or their reverse (if lane reversal is supported). The LTSSM proceeds to **Configuration.Complete**. **Partial Match**: The Lane Numbers don't match (or their reverse), but a subset of lanes can form a link. The downstream port reassigns Lane Numbers and returns to **Configuration.Lanenum.Wait**. **No Link Possible**: If no valid link can be formed, the LTSSM returns to **Detect**.
- **Lane Reversal**:
- If the upstream port supports lane reversal (e.g., remapping 0-3 to 3-0), it adjusts internally and responds with matching Lane Numbers.
- If it doesn't, it may send reversed Lane Numbers, requesting the downstream port to handle the reversal if capable.

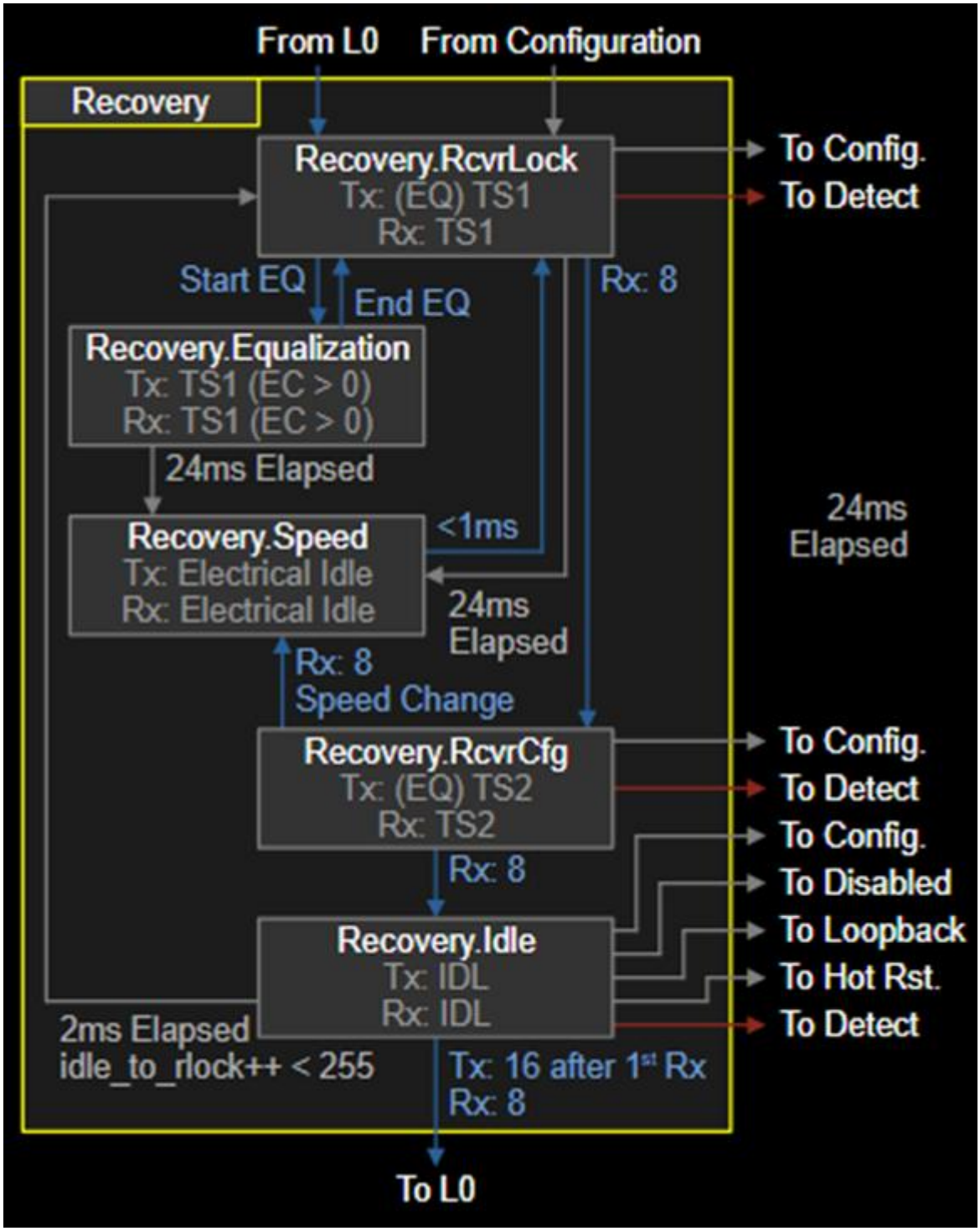
Configuration.Complete

- **Purpose**: Confirms the agreed link and lane configuration using **TS2 Ordered Sets**, a different sequence from TS1s, signaling the end of negotiation.
- **Transmitter (Both Ports)**: Sends **TS2 Ordered Sets** with: The agreed **Link Number**. The finalized **Lane Numbers**.
- **Receiver**: Listens for TS2s that match the transmitted Link and Lane Numbers.
- **Transition to Configuration.Idle**: Occurs when: At least **16 TS2s** have been sent after receiving one TS2, on **all lanes**. **Eight consecutive TS2s** are received with matching Link and Lane Numbers, on **all lanes**.
- **Timeout**: If the conditions are not met within **2ms**, the LTSSM returns to **Detect**.

Configuration.Idle

- **Purpose**: Verifies that the link is ready for normal operation by testing **scrambling**—a data encoding technique used in PCIe—using **Idle data symbols (IDL)**.
- **Transmitter (Both Ports)**: Sends **IDL symbols** on all configured lanes. Unlike TS1/TS2, these are scrambled.
- **Receiver**: Listens for IDL symbols from the link partner.
- **Transition to L0**: Occurs when: At least **16 consecutive IDL symbols** have been sent after receiving one IDL, on **all lanes**. **Eight consecutive IDL symbols** are received, on **all lanes**.
- **Timeout**: If the conditions are not met within **2ms**, the LTSSM returns to **Detect**.

Recovery substate



Recovery.RcvrLock

- **Purpose**: Synchronizes the transmitter and receiver during recovery, ensuring both are aligned for speed changes or post-equalization confirmation.
- **Occurrences**: Entered three times: **From L0 at 2.5GT/s**: **Transmitter**: Sends **TS1s** with the **Speed Change bit set** (1), optionally including equalization (EQ) settings like Transmitter Preset and Receiver Preset Hint for 8GT/s. **Receiver**: Listens for TS1s or TS2s with the Speed Change bit set. **Exit**: To **Recovery.RcvrCfg** when eight

consecutive TS1s or TS2s with Speed Change bit = 1 are received on all lanes. **From Recovery.Speed at 8GT/s**: After the speed change to 8GT/s, the LTSSM immediately proceeds to **Recovery.Equalization** to handle equalization at the new speed. **From Recovery.Equalization at 8GT/s**: **Transmitter**: Sends **TS1s** with the **Speed Change bit cleared** (0), **EC bits = 2'b00**, and current equalization settings (e.g., Transmitter Preset, Cursor Coefficients). **Receiver**: Listens for TS1s or TS2s with Speed Change and EC bits cleared. **Exit**: To **Recovery.RcvrCfg** when eight consecutive TS1s or TS2s with Speed Change bit = 0 are received on all lanes.

Recovery.RcvrCfg

- **Purpose**: Confirms readiness for the speed change or finalizes recovery at the new speed.
- **Occurrences**: Entered twice: **From Recovery.RcvrLock at 2.5GT/s**: **Transmitter**: Sends **TS2s** with the **Speed Change bit set** (1), optionally including EQ settings (Transmitter Preset, Receiver Preset Hint). **Receiver**: Listens for TS2s with the Speed Change bit set. **Exit**: To **Recovery.Speed** when eight consecutive TS2s with Speed Change bit = 1 are received on all lanes. **From Recovery.RcvrLock at 8GT/s**: **Transmitter**: Sends **TS2s** with the **Speed Change bit cleared** (0). **Receiver**: Listens for TS2s with the Speed Change bit cleared. **Exit**: To **Recovery.Idle** when eight consecutive TS2s with Speed Change bit = 0 are received on all lanes.

Recovery.Speed

- **Purpose**: Executes the physical speed change (e.g., from 2.5GT/s to 8GT/s) and prepares the link for operation at the new rate.
- **Actions**: **Transmitter**: Enters **electrical idle**, changes to the new speed, and configures equalization parameters (e.g., via phy_rate and phy_txeq_X signals). **Receiver**: Waits for all lanes to enter electrical idle. **Exit**: Returns to **Recovery.RcvrLock** after a wait of at least **800ns** (up to 1ms) once all receiver lanes are idle.
- **Fallback**: If the link fails to reestablish at the new speed, this substate may be re-entered to revert to the previous speed (e.g., 2.5GT/s).

Recovery.Equalization

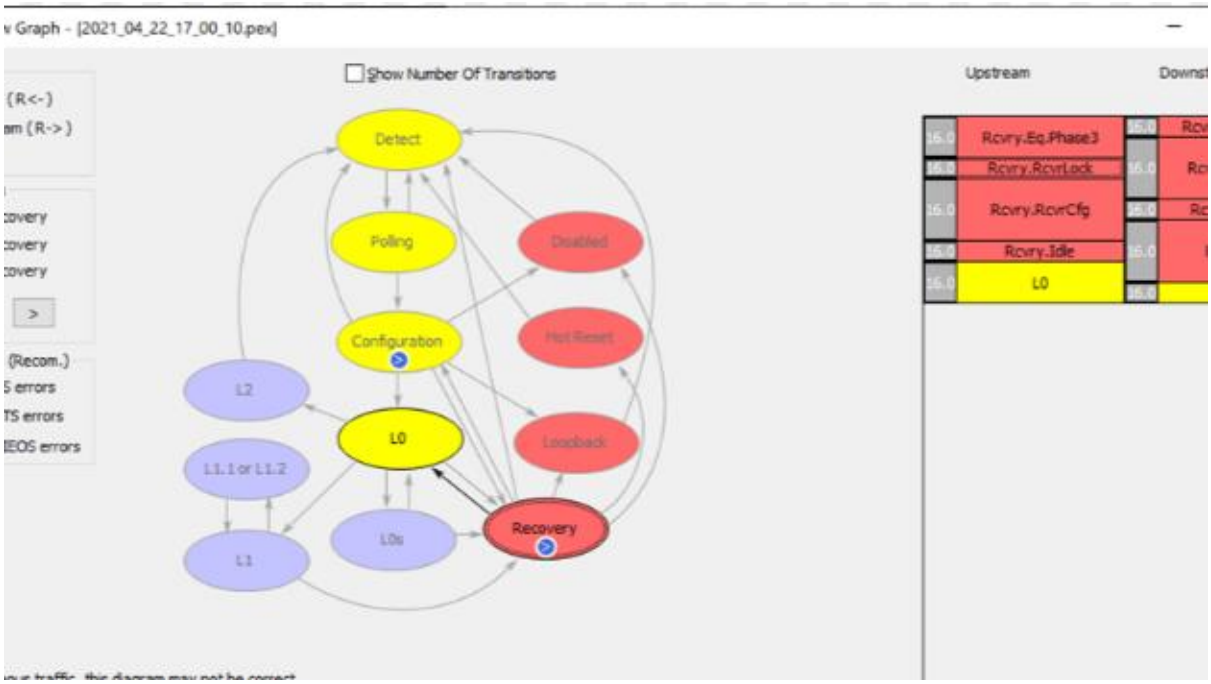
- **Purpose**: Optimizes signal quality at the new speed (e.g., 8GT/s) through transmitter equalization adjustments.
- **Phases**: **Phase 1** (always used): **Transmitter**: Sends **TS1s** with **EC = 2'b01**, including equalization settings (Transmitter Preset, Full Scale (FS), Low Frequency (LF), Post-Cursor Coefficient). **Receiver**: Listens for TS1s with EC = 2'b01. **Exit**: To **Recovery.RcvrLock** when two consecutive TS1s with EC = 2'b01 are received,

indicating acceptable settings. **Timeout**: If not met within **24ms**, returns to **Recovery.Speed** to revert to the lower speed. **Phases 2 and 3** (optional): Allow iterative tuning of equalization settings if initial presets are insufficient, evaluated via implementation-specific methods.

Recovery.Idle

- **Purpose**: Verifies link stability and scrambling functionality at the new speed (e.g., 8GT/s) before resuming normal operation.
- **Actions**: **Transmitter**: Sends **Idle data symbols (IDL, 8'h00)** on all configured lanes, scrambled at 8GT/s. **Receiver**: Listens for scrambled IDL symbols. **Exit**: To **L0** when: At least **16 consecutive IDLs** are transmitted after receiving one IDL on all lanes. **Eight consecutive IDLs** are received on all lanes. **Timeout**: If conditions aren't met within **2ms**, returns to **Detect**.

Test Plan for PCIe Physical layer:



The test plan verifies the key functionalities of the PCIe PHY layer, including the Logical Sub-block, Electrical Sub-block, encoding schemes, framing, scrambling, multi-lane link operation, and the LTSSM, ensuring reliable operation under normal and error conditions across supported data rates and link widths.

1. Logical Sub-block Tests

1.1 Transmit Section

Test Case	Description	Expected Result
Encoding for 2.5/5.0 GT/s (8b/10b)	Verify 8b/10b encoding of data from DLL for 2.5 GT/s.	8-bit data maps correctly to 10-bit symbols with DC balance.
Encoding for 8.0 GT/s (128b/130b)	Verify 128b/130b encoding for 8.0 GT/s with sync headers.	130-bit blocks (2-bit header + 128-bit payload) are correct.
Framing of TLPs	Verify TLP framing with STP start and END (or EDB) end symbols.	TLPs are framed with STP at start and END/EDB at end.
Framing of DLLPs	Verify DLLP framing with SDP start and END end symbols.	DLLPs are framed with SDP at start and END at end.
Scrambling of Data Symbols	Verify scrambling of data symbols using LFSR (per encoding scheme).	Data is scrambled; special symbols remain unscrambled.
Logical Idle Handling	Verify transmission of scrambled 0x00 (IDL) during idle periods.	Scrambled idle data maintains link synchronization.

1.2 Receiver Section

Test Case	Description	Expected Result
Symbol Identification	Verify identification of special symbols (e.g., COM, STP, SDP, END).	Symbols are correctly identified from incoming data.
Decoding for 2.5 GT/s (8b/10b)	Verify decoding of 10-bit symbols to 8-bit data for 2.5 GT/s.	Data is accurately decoded with no errors.
Decoding for 8.0 GT/s (128b/130b)	Verify decoding of 130-bit blocks for 8.0 GT/s.	Payload is correctly extracted from 130-bit blocks.
Error Checking	Introduce symbol errors and verify detection before forwarding to DLL.	Errors are detected and reported appropriately.
Framing Removal	Verify removal of STP, SDP, END, EDB symbols before passing data to DLL.	Data to DLL is free of framing symbols.

2. Electrical Sub-block Tests

Test Case	Description	Expected Result
Differential Signaling	Verify TX/RX pairs for differential signaling integrity.	Signals are transmitted/received with noise resistance.
Impedance Matching	Verify 100Ω termination on all lanes.	Termination resistors match 100Ω specification.
Signal Integrity	Verify equalization and EMI reduction techniques for signal quality.	Signals maintain integrity with minimal distortion.
Clock Recovery	Verify Clock Data Recovery (CDR) synchronizes data sampling.	Clocks are recovered accurately from the data stream.
Standards Compliance	Verify voltage and jitter compliance with PCIe specifications.	Signals meet PCIe voltage/jitter requirements.
SerDes Functionality	Verify serialization (TX) and deserialization (RX) of data.	Data is correctly serialized and deserialized.
Calibration and Self-Test	Verify self-test mechanisms for signal quality calibration.	Calibration completes successfully, ensuring quality.

3. Framing and Special Symbols Tests

Test Case	Description	Expected Result
COM Symbol Usage	Verify COM (K28.5) initializes lanes and links.	COM symbols correctly initialize link operation.
STP and SDP Framing	Verify STP (TLP) and SDP (DLLP) mark packet starts.	Packets start with correct STP or SDP symbols.
END and EDB Usage	Verify END terminates packets; EDB marks nullified TLPs.	Packets end with END; nullified TLPs end with EDB.
SKP for Clock Compensation	Verify SKP (K28.0) adjusts for clock differences.	SKP symbols are inserted/removed as needed.
Logical Idle Handling	Verify IDL symbol transmission during idle periods for 8.0 GT/s+.	IDL symbols maintain synchronization during idle.

4. Multi-Lane Link Operation Tests

Test Case	Description	Expected Result
Symbol Striping	Verify symbol distribution across lanes (e.g., x4: byte 0 to Lane 0, etc.).	Symbols are correctly striped across all lanes.
Framing Symbol Placement	Verify STP/SDP start on Lane 0 (or 4*N for wider links) from Logical Idle.	Framing symbols align with specified lanes.
PAD and IDL Handling	Verify PAD/IDL fill unused lanes in wider links after END/EDB.	Unused lanes are filled with PAD or IDL symbols.

5. LTSSM Tests

5.1 State Transition Tests

Test Case	Description	Expected Result
Detect to Polling	Verify transition upon detecting receiver termination.	LTSSM moves to Polling when a link partner is detected.
Polling to Configuration	Verify transition after 8 consecutive TS1/TS2 with PAD for Link/Lane numbers.	LTSSM moves to Configuration after synchronization.
Configuration to L0	Verify transition after link width and lane negotiation.	LTSSM enters L0 for normal operation.
L0 to Recovery	Verify transition on error or speed change request.	LTSSM enters Recovery to retrain the link.
Recovery to L0	Verify successful retraining and return to L0.	Link returns to L0 after retraining.
Gen1 to Gen3 Speed Change	Initiate speed change from Gen1 to Gen3 during Recovery.	LTSSM enters Recovery, retrains at Gen3 speed with equalization, returns to L0.
Gen3 to Gen4 Speed Change	Initiate speed change from Gen3 to Gen4 during Recovery.	LTSSM enters Recovery, retrains at Gen4 speed with advanced equalization (e.g., CTLE/DFE), returns to L0.
Gen4 to Gen5 Speed Change	Initiate speed change from Gen4 to Gen5 during Recovery.	LTSSM enters Recovery, retrains at Gen5 speed with precision equalization (e.g., FIR/FFE taps), returns to L0.
Direct Gen1 to Gen5 Speed Change	Force direct speed change from Gen1 to Gen5 during Recovery.	LTSSM enters Recovery, negotiates speeds Gen5, applies Gen5 equalization, and settles at Gen5 returns to L0.
L0 to L0s/L1/L2	Verify transitions to low-power states during idle (EIOS sent/received).	Link enters L0s, L1, or L2 correctly.
Loopback Mode	Verify entry into Loopback; receiver retransmits data (adjusts SKP if needed).	Link operates in Loopback, echoing data correctly.
Hot Reset	Verify link reset via TS1/TS2 without power cycle.	Link reinitializes successfully after Hot Reset.
Disabled State	Verify entry on critical failure or software command.	Link enters Disabled, requiring intervention.

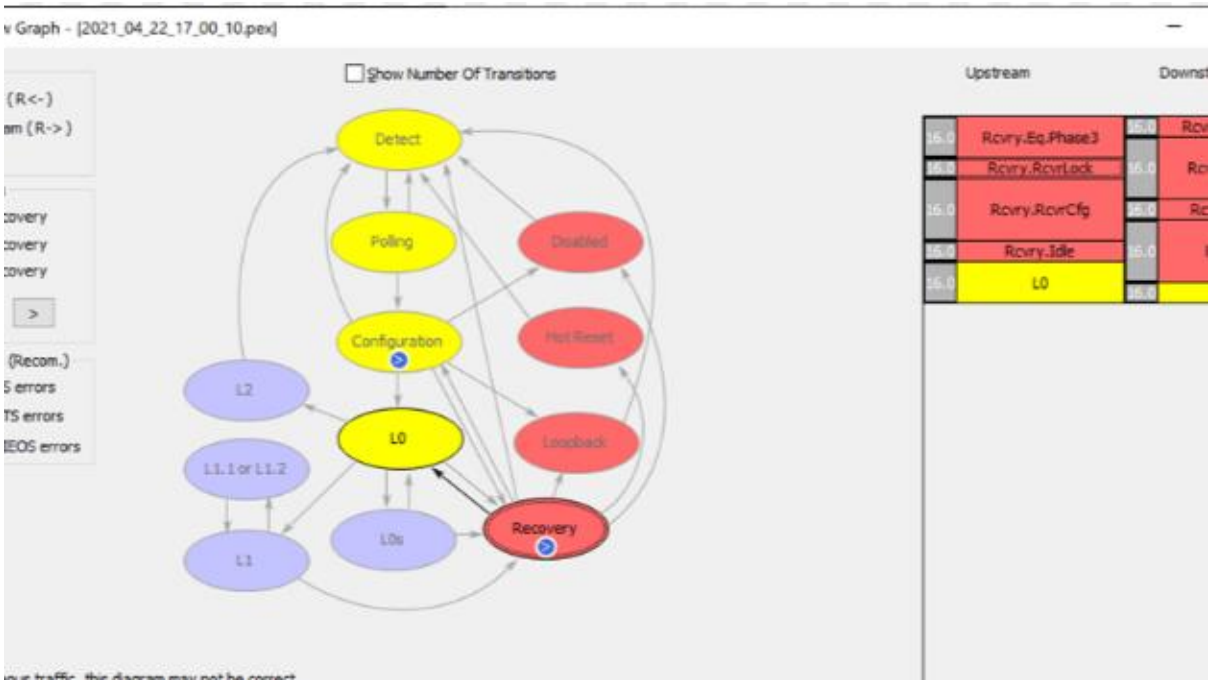
6. Negative Tests

Test Case	Description	Expected Result
Symbol Error Injection	Introduce errors in received symbols; verify detection and recovery.	Errors detected; LTSSM enters Recovery or Detect.
Lane Failure Simulation	Disable one or more lanes; verify link width negotiation.	Link negotiates reduced width or returns to Detect.
Timeout Conditions	Force timeouts (e.g., 24ms in Polling); verify state fallback.	LTSSM returns to Detect or appropriate state.
Unsupported Data Rates	Request unsupported rate (e.g., 64 GT/s); verify fallback to supported rate.	Link operates at highest mutually supported rate.

Test Plan Notes

- **Test Setup**: Assumes a PCIe link between a Root Complex and an Endpoint with configurable data rates (2.5, 5.0, 8.0, 16.0, 32.0 GT/s) and link widths (x1, x4, x8, x16).
- **Execution**: Tests should be repeated across all supported data rates and link widths to ensure full coverage.
- **Dependencies**: Requires interaction with the Data Link Layer for data exchange and LTSSM state directives.
- **Error Handling**: Negative tests verify robust error detection and recovery mechanisms.
- **Compliance**: External test equipment is needed for electrical compliance verification.

Verification: Identifying and Resolving PCIe PHY Errors



Pin Connection Error Scenarios

The PCIe PHY connects to other components via the PIPE (PHY Interface for PCIe) interface, and incorrect pin connections can lead to significant issues. Below five common error-prone scenarios:

1. **Ambiguity in Data Pin Declaration** For a 16-bit PIPE interface with four lanes, the data pin can be declared as a single 64-bit pin or four 16-bit wires. If declared as a 64-bit pin, it's unclear whether bits [15:8] represent the second byte of the first lane or the first byte of the second lane. A wrong connection could result in the link using fewer lanes than intended, requiring extensive debugging later.
2. **Signal Width Variations Across PIPE Versions** The TxDeemph signal, which controls signal distortion, is 18 bits in the latest PIPE specification but was 1 bit in earlier versions. Engineers might mistakenly leave it unconnected or improperly combine the 18 bits, causing signal integrity issues.
3. **Shared vs. Per-Lane Signals** The PowerDown signal is 3 bits in the latest specification (previously 2 bits) and may be shared or per-lane depending on the design. Misdeclaring it as a segregated signal for all lanes can disrupt link-up. Similarly, the Rate signal (2 bits now, 1 bit previously) can cause issues if misconfigured.
4. **Clock Connection Problems** The PHY can generate its clock, or the testbench can supply it externally (e.g., GEN1 clock). Errors occur if the clock connection is missed, duplicated, or not updated during speed changes (e.g., sticking to GEN1 clock rates), leading to synchronization failures.

5. **Reset Connection Issues** Common mistakes include leaving the reset unconnected, keeping it always on, or setting its duration too short or too long. These errors can prevent the PHY from initializing correctly.

Configuration Error Scenarios

Correctly configuring the PHY and its Verification IP (VIP) is crucial, but errors here can stall verification. Five examples are provided:

1. **Clock Frequency vs. PIPE Width Mismatch** If the VIP is set for a clock frequency transition while the PHY supports PIPE width changes during speed shifts, a deadlock can occur, halting communication.
2. **Missing Initial PIPE Width** Forgetting to set the initial PIPE width for link-up can prevent the connection from establishing properly.
3. **Double Scrambling Conflict** The Media Access Controller (MAC) typically scrambles data on the PIPE interface. If the PHY is also configured to scramble, the receiving end gets unrecognizable data due to redundant scrambling.
4. **Loopback Master Misconfiguration** In loopback testing, only the VIP on the PHY side should be the slave, while the serial-side VIP is the master. Configuring both as masters disrupts the loopback state.
5. **Incorrect LTSSM Timer Values** The Link Training and Status State Machine (LTSSM) timers should follow protocol specs, but adjusting them for faster simulation without matching PHY requirements can cause timing issues.

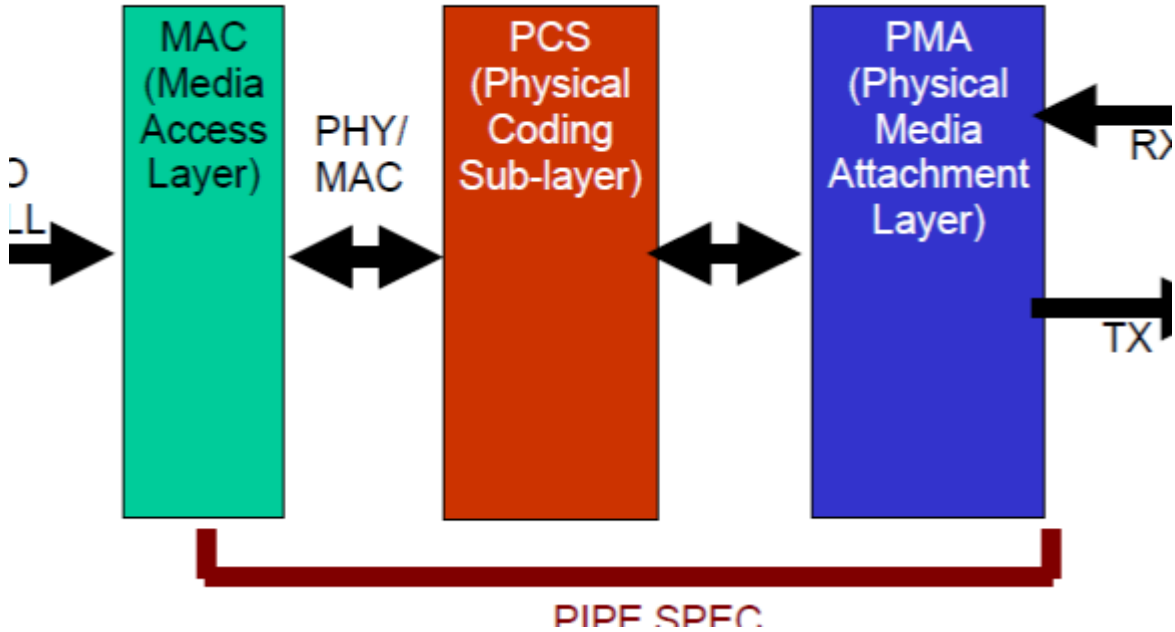
Link-Up Error Scenarios

After pin connections and configurations are set, achieving a successful link-up (initial communication between devices) is the next hurdle. Eight common issues are listed:

1. **No Receiver Detection** If the PHY sets the Phystatus signal low, the MAC may not initiate receiver detection.
2. **Need for Retry** The PHY might require another detection attempt, which could be missed if not configured.
3. **Reset-Related Non-Response** A misconfigured reset prevents the PHY from responding to detection attempts.
4. **PowerDown Signal Ignored** The PHY fails to react to changes in the PowerDown signal, stalling link-up.
5. **RxElecIdle Timing Error** Incorrect timing of the RxElecIdle signal can truncate valid packets mid-transmission.

6. **Equalization Failure** During initial equalization, incorrect coefficient values stop the PHY from responding.
7. **Rate Signal Non-Response** The PHY doesn't adjust to a speed change signaled by the MAC's Rate signal.
8. **Short Timer in Recovery** A short timer value prevents the PHY from completing a speed change in recovery mode.

PCIe: Single Root I/O Virtualization



SR-IOV as a technology to enhance hardware resource utilization by allowing multiple System Images (SIs) to share PCI hardware resources.

Here's a summary of the key aspects covered:

Problem addressed: Traditional virtualization using a Virtualization Intermediary (VI) can lead to performance degradation for I/O intensive workloads due to the VI intercepting and processing every I/O operation.

Traditional virtualization, without SR-IOV, relies on a Supervisor (Virtual Machine Manager or VMM) to emulate virtual devices, leading to I/O bottlenecks and performance limitations

SR-IOV Solution: SR-IOV aims to reduce this overhead by enabling the elimination of VI involvement in main data movement actions like DMA, memory space access, and interrupt processing, leading to significant performance improvements.

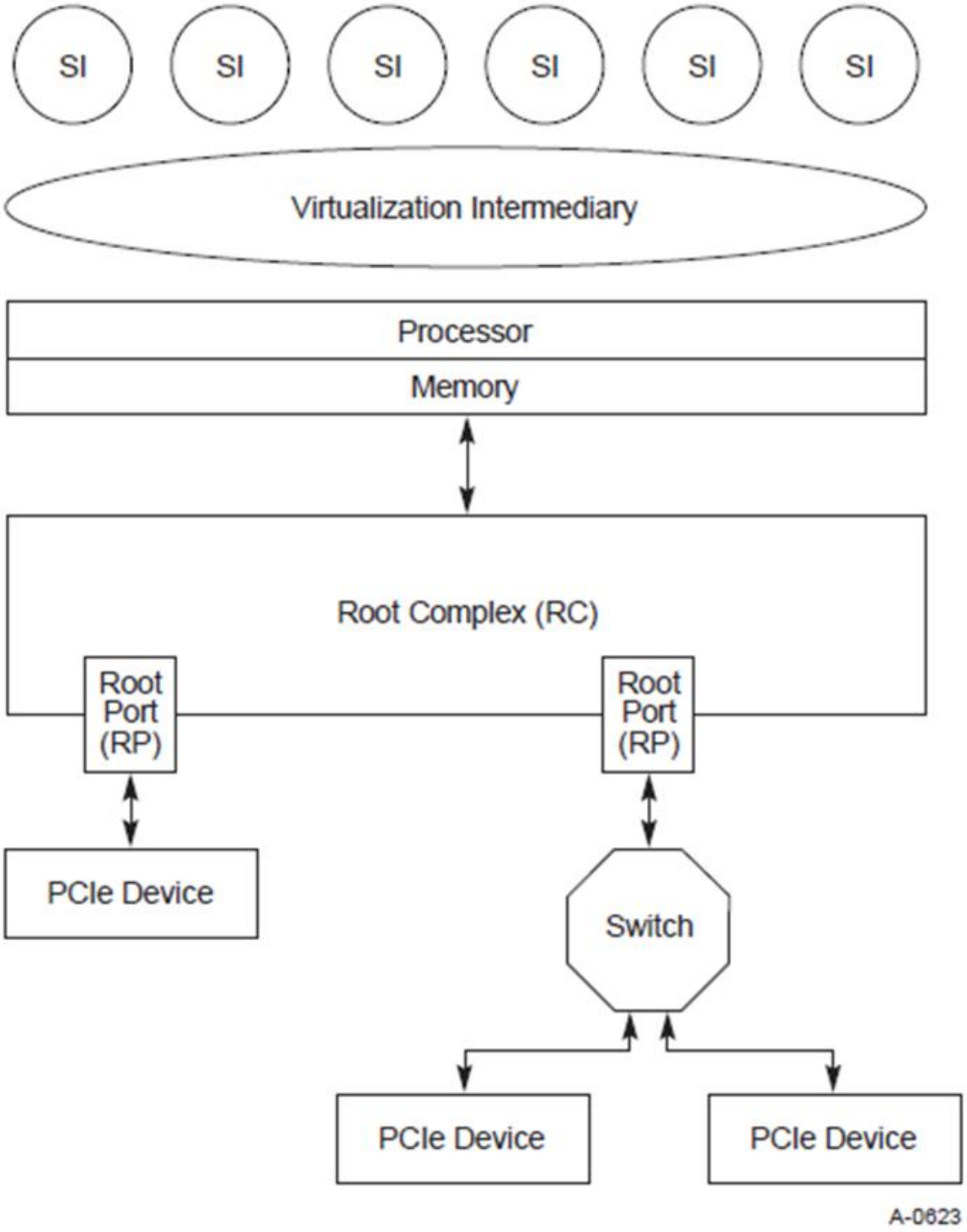
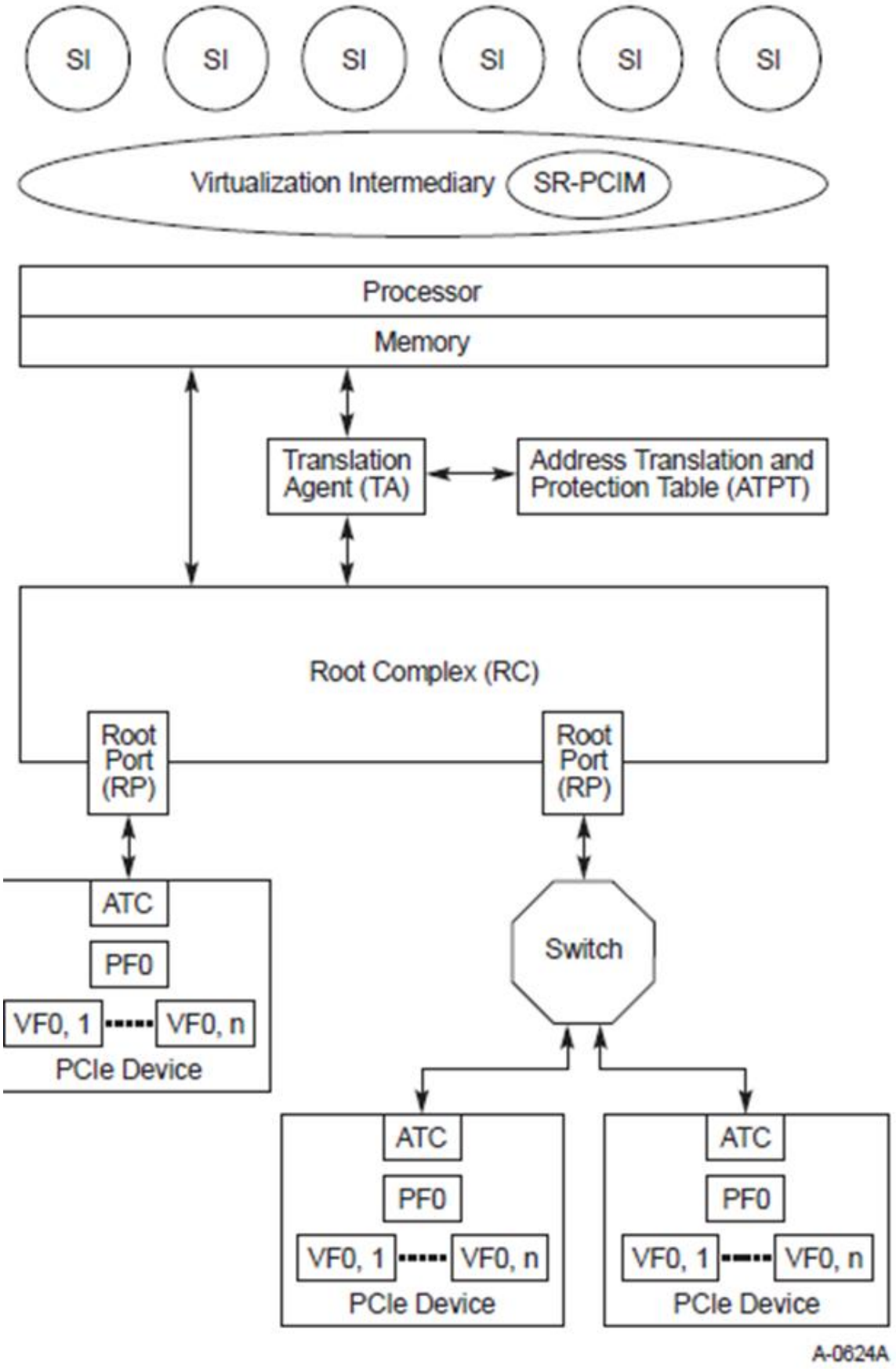


Figure 9-2 Generic Platform Configuration with a VI and Multiple SI



9-3 Generic Platform Configuration with SR-IOV and IOV Enablers

Benefits of SR-IOV:

o Eliminates VI involvement in main data movement.

o Provides a standardized method for resource configuration and management through Single Root PCI Manager (SR-PCIM).

o Reduces hardware requirements and costs associated with provisioning multiple I/O Functions.

o Allows integration with other I/O virtualization technologies for more complete solutions.

Key Functional Elements in SR-IOV:

SR-PCIM: Software responsible for configuring SR-IOV, managing Physical Functions (PFs) and Virtual Functions (VFs), and handling related events.

Translation Agent (TA) and Address Translation and Protection Table (ATPT): Optional hardware/software for translating PCIe transaction addresses into platform physical addresses, with ATPT storing translation data and TA potentially including an ATC for speed.

Address Translation Cache (ATC): Optional cache within the TA or PCIe Device to accelerate address lookups, improving performance by caching translations.

Access Control Services (ACS): Optional, controls TLP routing to block or redirect traffic, preventing unauthorized communication between Functions or SIs, enhancing security.

Physical Function (PF):

Full-featured PCIe functions that are managed and can control the PCIe device. A device can have up to eight PFs. A full PCIe Function that supports SR-IOV and is managed by SR-PCIM, a VI, or an SI.

Virtual Function (VF):

A lightweight PCIe Function (designed solely for data movement) directly accessible by an SI, with restricted configuration resources controlled by a trusted component. VFs associated with a PF must be of the same device type. each attached to a PF. A PF can have zero or more VFs, and different PFs on the same device can have varying numbers of VFs.

Comparison with multi-function PCIe Devices

- **Traditional Multi-Function Devices (Figure 9-4)**: Support up to 8 Functions per device, each with unique configuration and physical resources, managed through Function 0. With ARI, this can extend to 256 Functions, but resource scaling increases hardware costs.
- **SR-IOV Devices (Figure 9-5, 9-6, 9-7, 9-8)**: Use a single PF with multiple VFs to reduce costs, sharing configuration fields to minimize hardware. Examples include: Single PF

with N VFs (Figure 9-5). Multiple PFs, each with VFs, varying counts (Figure 9-6). Devices using multiple Bus Numbers for large VF counts (Figure 9-7). Mixed Functions, PFs, and VFs (Figure 9-8), with Function 0 always present.

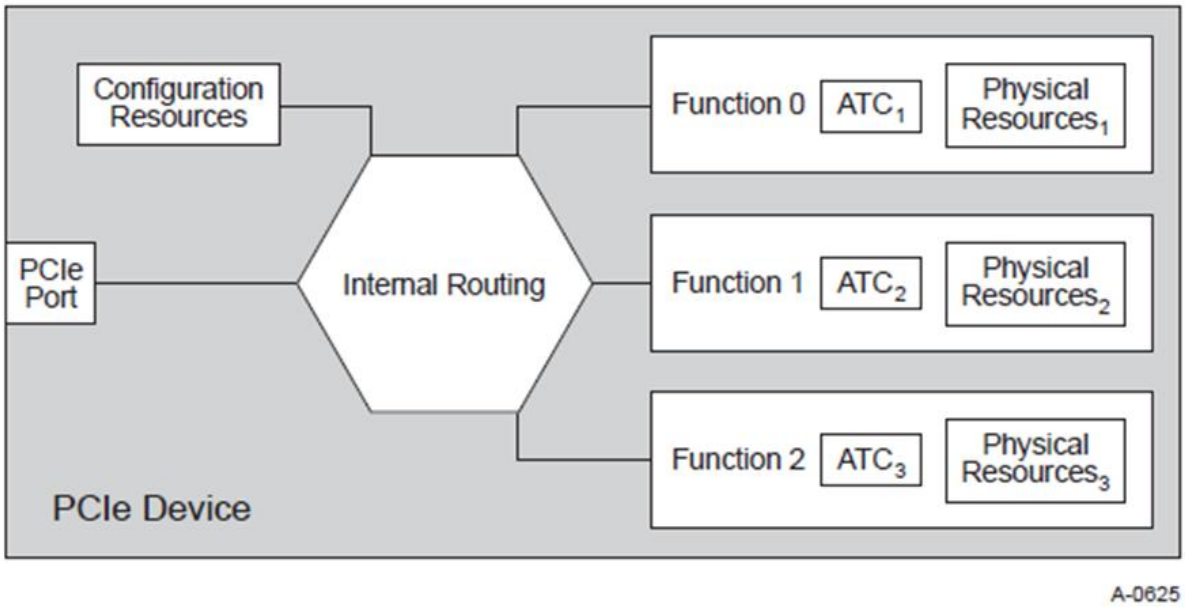


Figure 9-4 Example Multi-Function Device

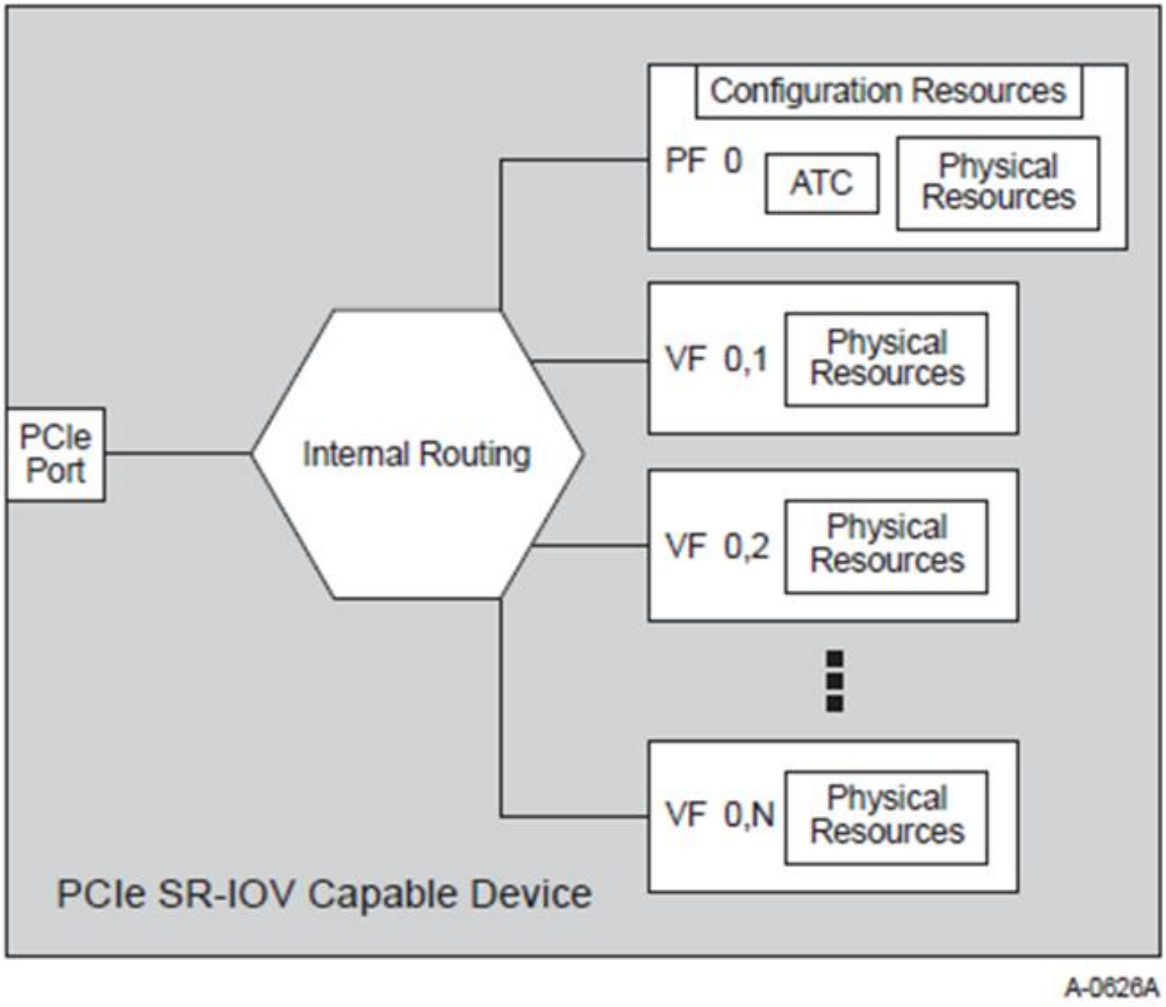


Figure 9-5 Example SR-IOV Single PF Capable Device

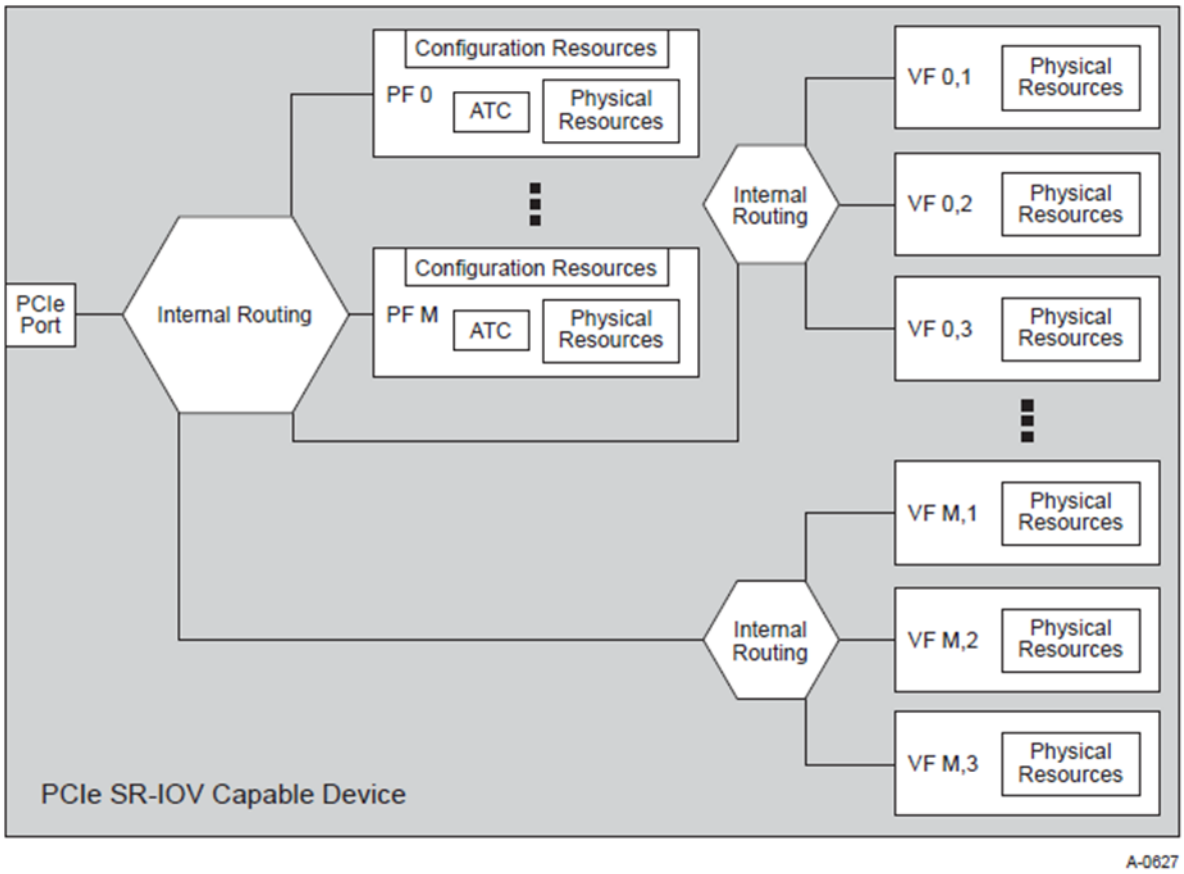


Figure 9-6 Example SR-IOV Multi-PF Capable Device

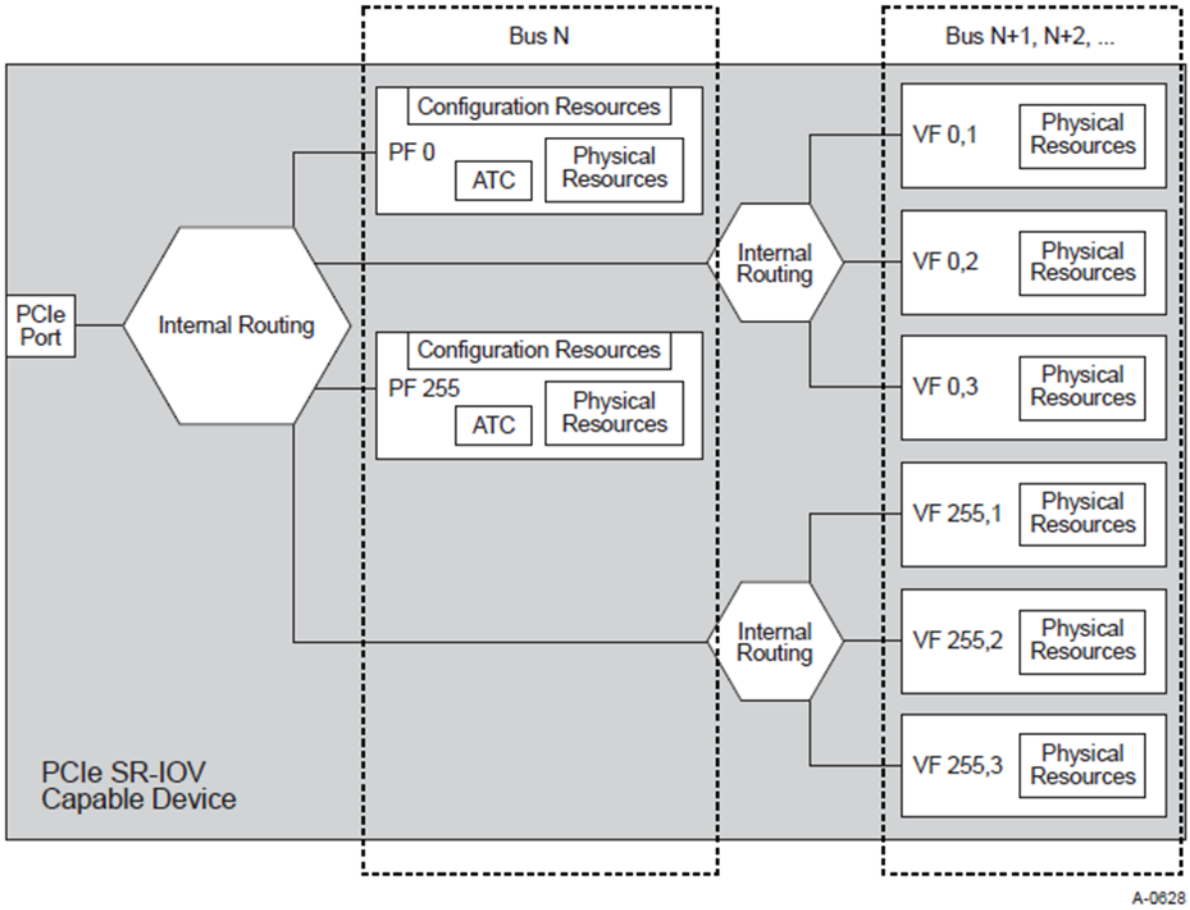


Figure 9-7 Example SR-IOV Device with Multiple Bus Numbers

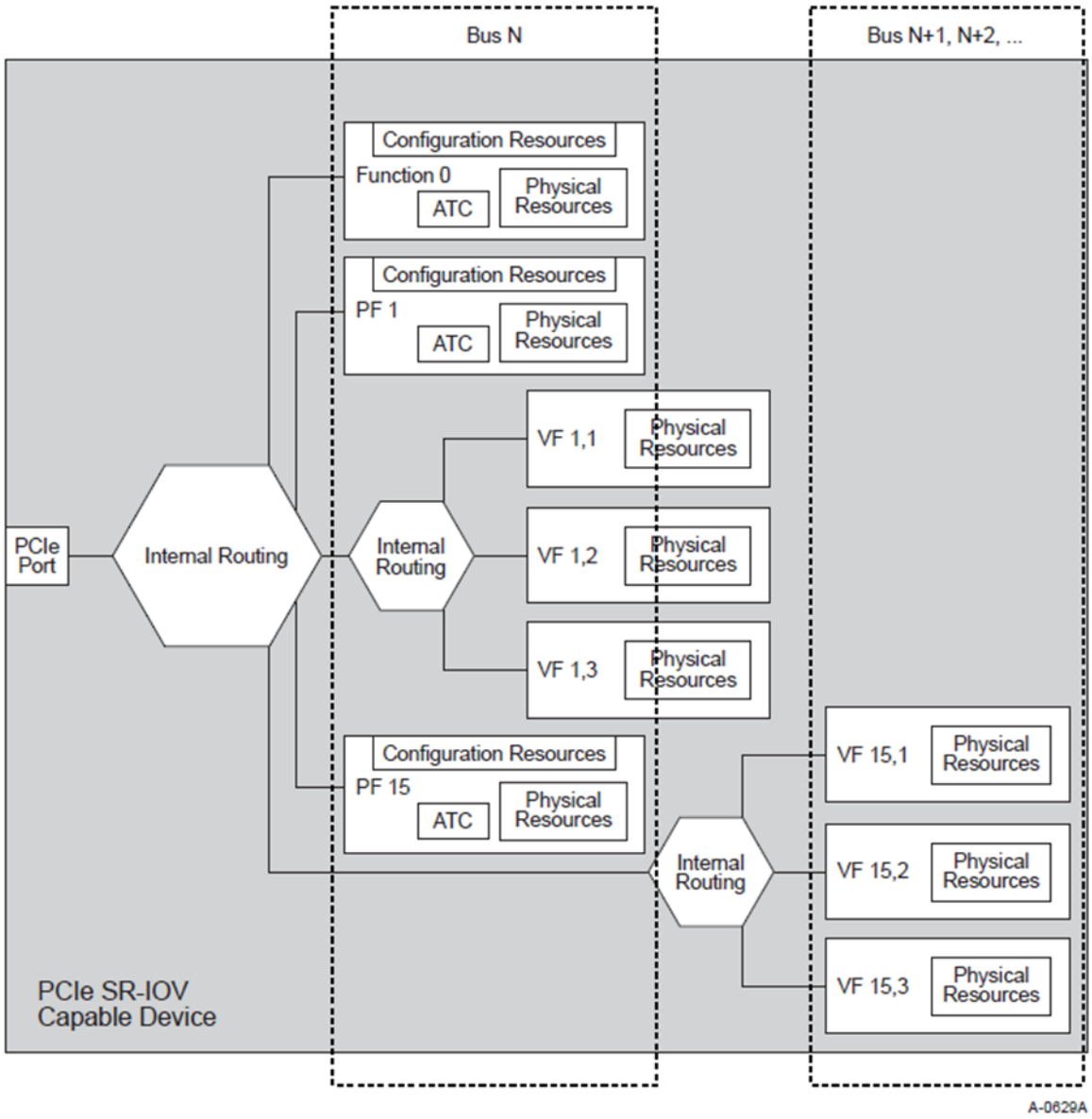


Figure 9-8 Example SR-IOV Device with a Mixture of Function Types

PCI Technologies Interoperability

SR-IOV builds on the PCI Express specification (PCIe 1.1 or later), ensuring compatibility:

- Does not affect the physical, data link, or transaction layers directly.
- Communicates capabilities via extended configuration space.
- Interoperates with PCI/PCI-X via bridges, allowing serial sharing by multiple SIs.
- Supports ATS and can mix SR-IOV with non-SR-IOV components in a hierarchy.

SR-IOV Initialization and Resource Allocation

- **Resource Discovery**: Software identifies SR-IOV capabilities by reading PF configuration space, enabling VFs via the VF Enable bit, and setting NumVFs.

- **VF BAR Mechanisms**: Configures memory space mapping, aligning VF BARs to system page boundaries using System Page Size, with no I/O Space support.
- **VF Discovery**: Calculates VF Routing IDs using PF Routing ID, First VF Offset, and VF Stride, ensuring unique IDs and handling multiple Bus Numbers if needed.

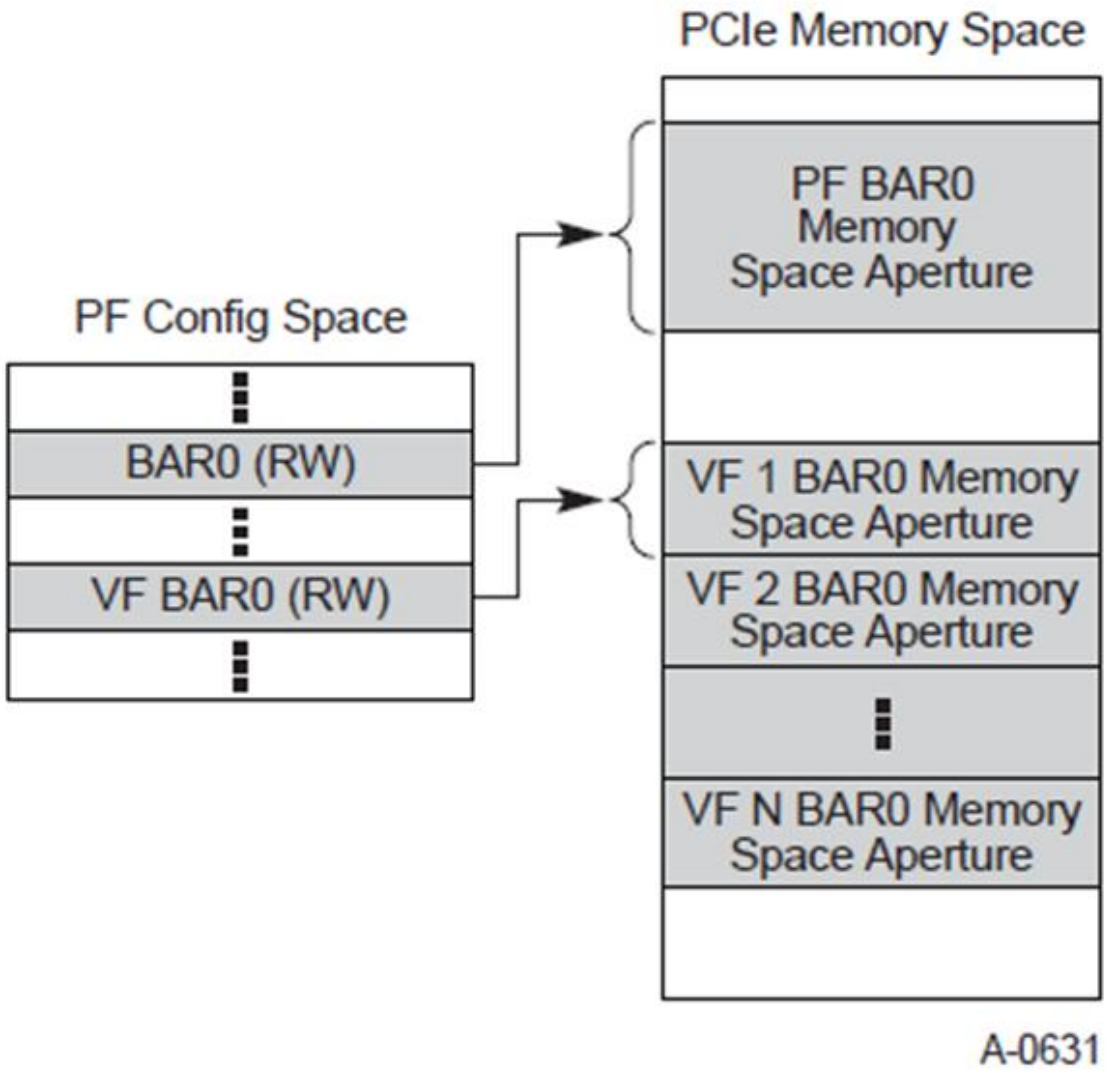


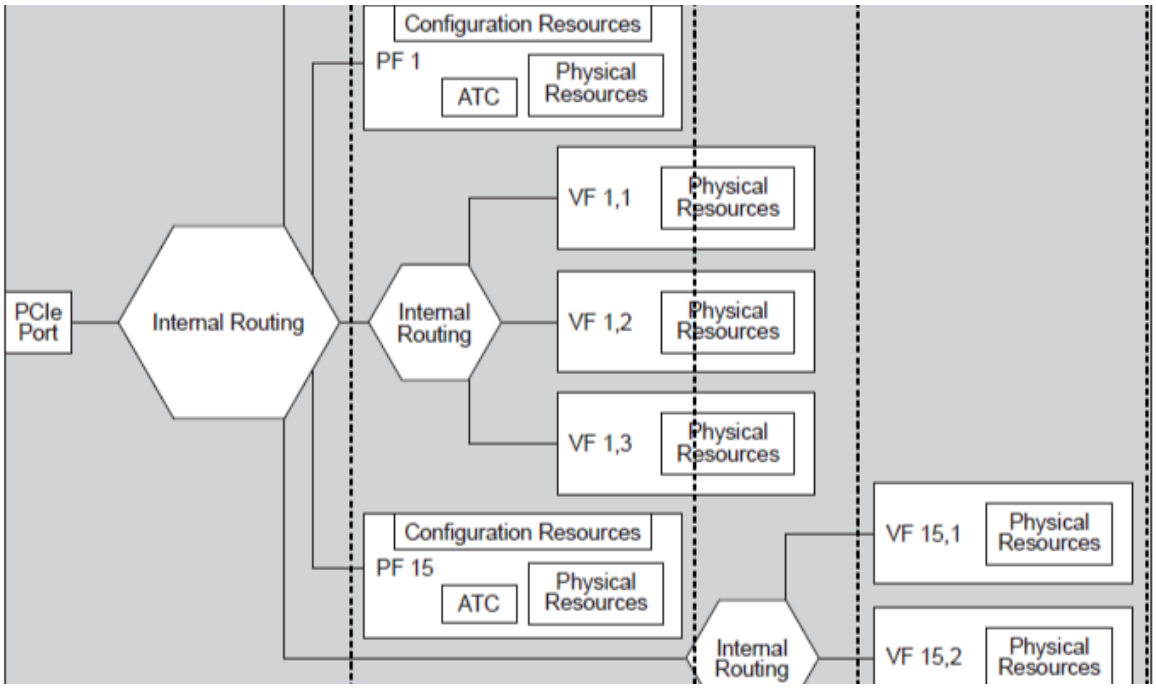
Figure 9-10 BAR Space Example for Single BAR Device

Reset Mechanisms

- **Conventional Reset**: Resets all Functions (PFs, VFs) to power-on state, clearing VF Enable, making VFs disappear.
- **Function Level Reset (FLR)**:
- **VF FLR**: Resets VF state without affecting existence; BARs and VF MSE remain unchanged.

- **PF FLR**: Resets PF state and SR-IOV capability, disabling VFs.

Verification issues: PCIe Data Link Layer

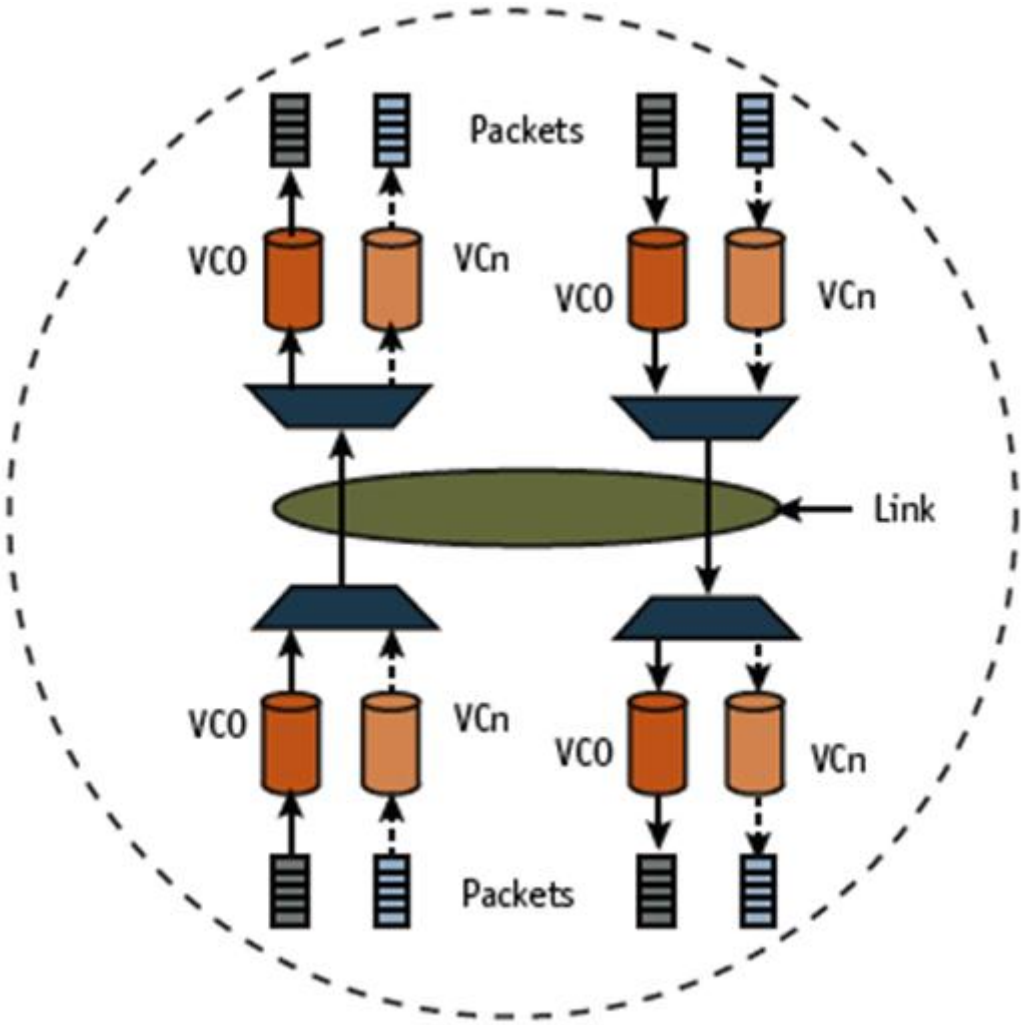


1. Flow Control (FC) Issues

The PCIe Data Link Layer uses a **Virtual Channel (VC)** mechanism combined with **Traffic Class (TC)** identification to support differentiated services and Quality of Service (QoS) for various applications. Flow control (FC) ensures that packets are transmitted only when the receiving end has enough buffer space (credits) to accept them. However, several problems in this area:

- **VC0 vs. VCx Behavior**: VC0 is the default virtual channel, and designers typically handle its credit flow correctly. As a result, the device-under-test (DUT) behaves properly for VC0. However, issues emerge with other virtual channels (denoted as VCx). Updated flow control information for VCx is not sent properly, meaning the receiving end doesn't know when buffer space is available. This leads to **starvation**, where requests on VCx are not serviced due to a lack of credits.
- **Message Packet Credit Handling**: Certain message packets, such as system SSPL (System-Specific Packet Latency) or vendor-defined messages, are not supported by the DUT and are simply ignored. The DUT fails to update flow control information for these packets, so credits associated with them are not freed up. This causes additional **starvation**, as the sender assumes the receiver's buffers are still occupied, halting further transmissions.

In summary, improper handling of FC updates for VCx and unsupported message packets disrupts credit flow, leading to inefficiencies and stalled operations.



2. DL_Inactive Status Issues

The Data Link Layer (DL) manages the state of the PCIe link, communicating with the transaction and physical layers. After a reset (hot, warm, or cold), the DL layer enters the **DL_Inactive** state, which is its initial state when the link is non-operational or nothing is connected. several design flaws in this process:

- **Improper Reset of DL Layer**: Upon entering DL_Inactive, all DL layer state information—such as **NEXT_TRANSMIT_SEQ** (next sequence number to transmit), **ACKD_SEQ** (last acknowledged sequence), **REPLAY_NUM** (replay attempt count), and **NEXT_RCV_SEQ** (next expected sequence to receive)—should reset to default values. The retry buffer, which stores packets for retransmission, should also be cleared. In some designs, this reset doesn't happen. As a result, these variables retain old values, and the retry buffer isn't emptied.
- **Packets Sent in DL_Down State**: When the link is re-established, the DL layer may still be in a transitional state like **FC_INIT1** (part of flow control initialization) or even **DL_Down** (indicating the link is not operational). Despite this, some designs allow the transaction layer to send packets from the application layer. Since the retry buffer

wasn't cleared and sequence numbers weren't reset, the first packet sent doesn't start with sequence number 0. Instead, old packets from the retry buffer are transmitted, leading to **incorrect sequence numbers** and **out-of-sync credit logic** between the sender and receiver.

These issues cause confusion in packet ordering and credit tracking, undermining reliable communication.

3. Replay Mechanism Issues

The DL layer ensures reliable packet delivery by checking the integrity of Transaction Layer Packets (TLPs) and requesting retransmission (replay) if errors are detected. A replay can be triggered by a **NAK** (negative acknowledgment) from the receiver or the expiration of the **REPLAY_TIMER**. two key problems:

- **Mixing New Packets with Replays**: During a replay, the DL layer should retransmit only the packets that failed. However, some designs fail to block new packets from the transaction layer. As a result, these new packets get interleaved with replayed packets, disrupting the intended sequence and potentially confusing the receiver.
- **Mishandling ACK/NAK During Replay**: Normally, an **ACK** (acknowledgment) confirms successful receipt, while a **NAK** requests retransmission. Some designs mishandle these signals during replay, retransmitting packets that were already acknowledged. This unnecessary retransmission wastes bandwidth and **degrades performance**.

These flaws in the replay mechanism compromise the efficiency and reliability of error recovery.

4. Lane Count Less Than Maximum Supported

PCIe links can operate with varying numbers of lanes (e.g., x1, x4, x8), depending on the hardware and negotiation. a maximum supported lane count of 8, but the link might establish with fewer lanes (e.g., 4). This mismatch causes timing-related issues:

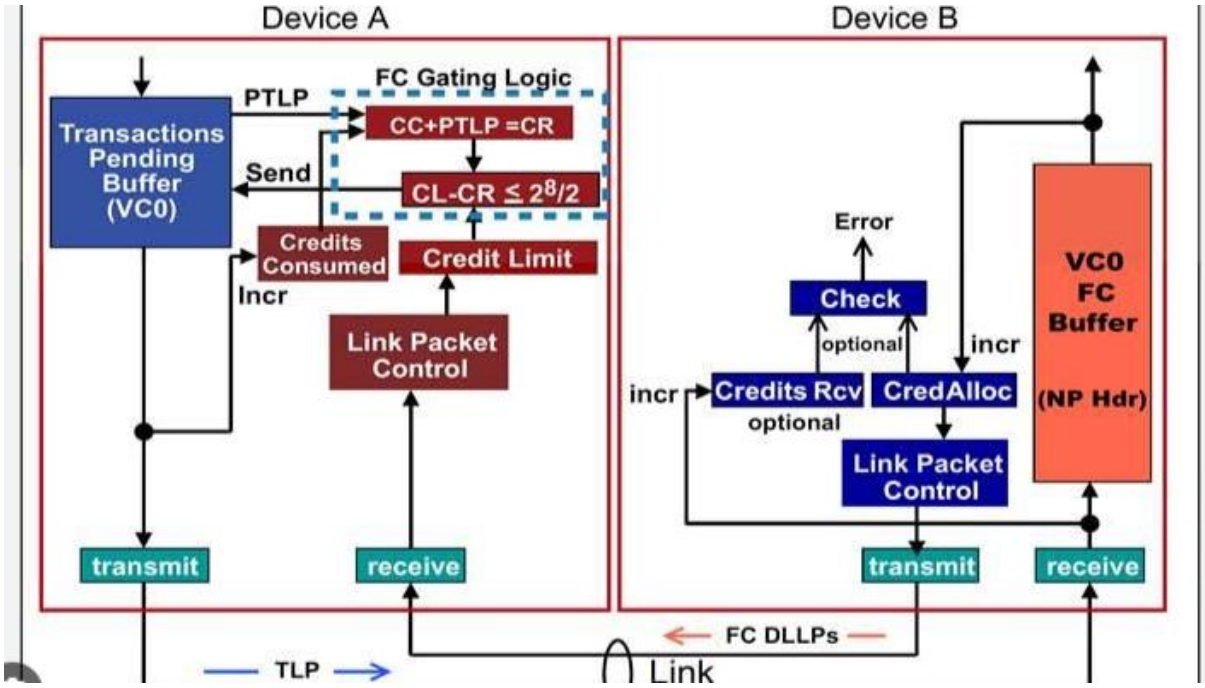
- **Timer Misconfiguration**: Timers like **AckNak_LATENCY_TIMER** (time to wait for an ACK/NAK) and **REPLAY_TIMER** (time before triggering a replay) are set based on the maximum lane count (8 lanes in this case). When the link operates with fewer lanes (e.g., 4), these timers don't adjust to the reduced bandwidth and latency characteristics of the actual configuration.
- **Impact**: The timers remain calibrated for a higher lane count, leading to delays or premature timeouts. This mismatch can reduce performance (e.g., slower error recovery) or affect reliability (e.g., replays triggered too early or too late).

Properly scaling these timers to the active lane count is critical for optimizing link operation.

Link Tests:

- Reserved fields – Device ignores them
- NAK response – Device will resend after receiving NAK
- Replay Timers – Device will resend packet if no response
- Replay Count - Device will resend multiple times when no response
- Link Retrain - Device will retrain if continued no response
- Replay TLP order – Device replays TLPs in proper order
- Bad CRC - Device detects, drops, and logs (DLLPs & TLPs)
- Undefined packet – Device ignores
- Bad Sequence Number – Device detects, drops, and logs
- Duplicate TLP - Device returns data once

PCIe Transaction Layer issues and TLP Errors:



Common Issues: Transaction Layer:

1. **Interpretation of the Read Completion Boundary (RCB) Parameter**:

· The RCB defines the address boundaries (e.g., 64 bytes or 128 bytes) at which a memory read request can be split into multiple completions, based on the request size and the Max_Payload_Size.

· **Issue**: Some endpoint designs misinterpret the RCB bit. For example, when the RCB is set to 0 (indicating 128-byte alignment), devices may incorrectly send completions aligned to 64 bytes.

· **Consequence**: The receiver treats these completions as malformed, leading to communication failures.

2. **Max_Read_Request_Size Register**:

· Located in the PCIe capability structure within the first 256 bytes of PCI configuration space, this register sets the maximum size for memory read requests.

· **Issue**: When misinterpreted, the register's value is incorrectly applied to the receive logic, causing valid read requests exceeding this size to be flagged as malformed.

· **Consequence**: This disrupts normal operation by rejecting properly formed requests.

3. **Implementing Advanced Error Reporting (AER)**:

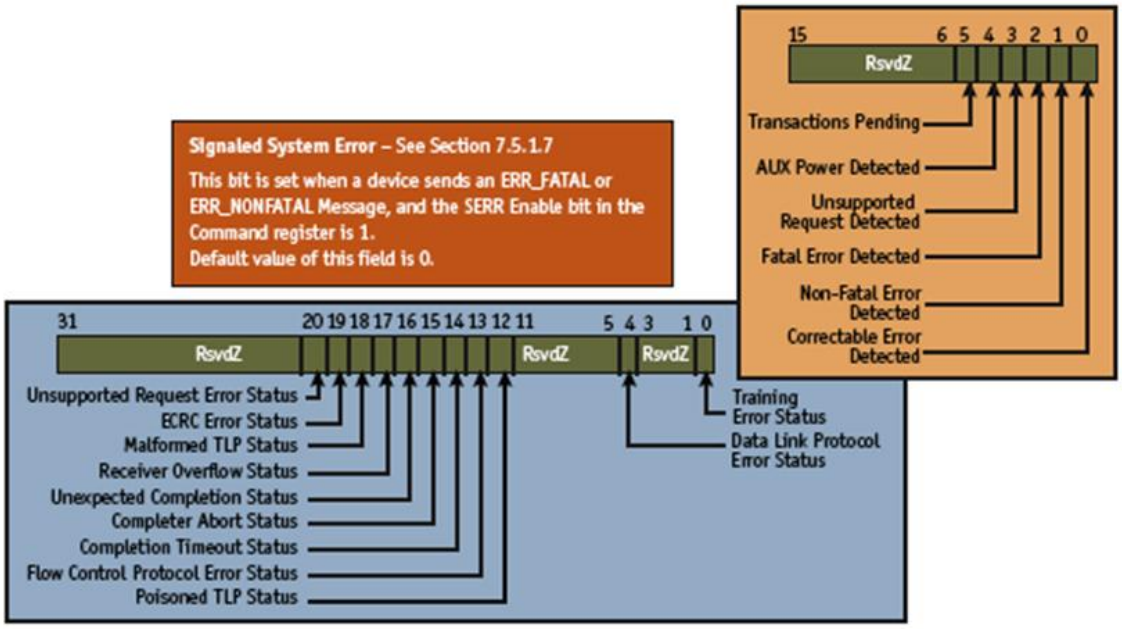
· PCIe supports two error reporting levels:

o **Baseline Error Reporting**: Mandatory for all devices, providing basic error detection.

o **Advanced Error Reporting (AER)**: Optional, offering detailed error logging via a specific capability structure.

· **Issue**: While AER registers are updated correctly, some implementations fail to update related standard PCI configuration space registers (e.g., status registers).

· **Consequence**: This leads to incomplete error reporting, reducing the system's ability to diagnose and recover from errors.



TLP Errors and it's handling:

1. Poisoned TLP Received

- **Type**: Uncorrectable (non-fatal)
- **Description**: This error occurs when a received TLP has its **EP (Error Poisoning)** bit set, indicating that the data within the packet is corrupted or invalid. Despite the corruption, the TLP is still delivered to the Application Layer.
- **Handling**: The Application Layer must decide how to respond to the poisoned TLP, such as discarding it, logging the error, or taking corrective action.

2. ECRC Check Failed

- **Type**: Uncorrectable (non-fatal)
- **Description**: This error occurs when the **End-to-End Cyclic Redundancy Check (ECRC)** fails, even though the TLP is not malformed and the Link CRC (LCRC) check passes. ECRC is an optional feature that provides an additional layer of data integrity verification.

- **Handling**: If the TLP is a **non-posted request** (e.g., a read), the Hard IP block generates a completion packet with **Completer Abort** status. In all cases, the TLP is deleted by the Hard IP block and not passed to the Application Layer.

3. Unsupported Request for Endpoints

- **Type**: Uncorrectable (non-fatal)
- **Description**: This error occurs when a device receives an unsupported or invalid request. Examples include: Type 0 Configuration Requests for non-existing functions. Completion transactions with a Requester ID that doesn't match the bus, device, and function number. Unsupported messages (e.g., invalid message types). Type 1 Configuration Request TLPs received from the PCIe link. Locked memory reads (MEMRDLK) or locked completion transactions on a native Endpoint. 64-bit memory transactions where the upper 32 bits of the address are zero. Memory or I/O transactions with no matching Base Address Register (BAR). Memory transactions when the Memory Space Enable bit (bit [1] in the PCI Command register at offset 0x4) is disabled. Poisoned configuration write requests (CfgWr0).
- **Handling**: The Hard IP block deletes the TLP and does not pass it to the Application Layer. For **non-posted requests**, the Hard IP block generates a completion with **Unsupported Request** status.

4. Completion Timeout

- **Type**: Uncorrectable (non-fatal)
- **Description**: This error happens when a request sent by the Application Layer does not receive a corresponding completion TLP within a predefined time limit. The Application Layer must implement a timeout detection mechanism.
- **Handling**: The Application Layer reports this error using the cpl_err[0] signal. It is responsible for managing the timeout and taking appropriate action.

5. Completer Abort

- **Type**: Uncorrectable (non-fatal)
- **Description**: This error occurs when the Application Layer aborts the processing of a received TLP due to an internal issue (e.g., an unresolvable error). It reports this using the cpl_err[2] signal.
- **Handling**: If the TLP is a **non-posted request**, the Hard IP block generates a completion with **Completer Abort** status.

6. Unexpected Completion

- **Type**: Uncorrectable (non-fatal)

- **Description**: This error is triggered when a completion TLP is received that doesn't correspond to any outstanding request. Examples include: The Requester ID in the completion doesn't match the Endpoint's configured ID. The completion packet has an invalid or out-of-range tag number (e.g., exceeding the specified tag limit). The tag doesn't match any pending request. The completion packet for an I/O or Configuration Space request has a length greater than 1 dword. The completion status is **Configuration Retry Status (CRS)** for a request not targeting Configuration Space.
- **Handling**: The Hard IP block deletes the unexpected completion TLP and does not pass it to the Application Layer. The Application Layer can report additional unexpected completion scenarios (e.g., mismatched completion lengths) using the cpl_err[2] signal.

7. Receiver Overflow

- **Type**: Uncorrectable (fatal)
- **Description**: This error occurs when a received TLP exceeds the allocated **Flow Control (FC)** credits for its type, leading to a buffer overflow. This indicates a serious violation of the flow control mechanism.
- **Handling**: The Hard IP block deletes the TLP and does not pass it to the Application Layer. As a fatal error, it may require system-level intervention.

8. Flow Control Protocol Error (FCPE)

- **Type**: Uncorrectable (fatal)
- **Description**: This error is triggered when a component fails to receive updated flow control credits within the required 200 µs time limit, violating the PCIe flow control protocol.
- **Handling**: The Hard IP block detects and reports this as a fatal error, which could lead to deadlock or starvation in the system.

9. Malformed TLP

- **Type**: Uncorrectable (fatal)
- **Description**: This error occurs due to various TLP formatting issues, such as: Data payload exceeding the maximum payload size. The **TD (TLP Digest)** field is set but no digest is present, or a digest exists but TD is not set. Violation of byte-enable rules (optional check by the Hard IP block). Mismatched type and length fields with the TLP's total length. Invalid format/type combinations per the PCIe specification. Address/length combinations causing memory access to cross a 4 KB boundary

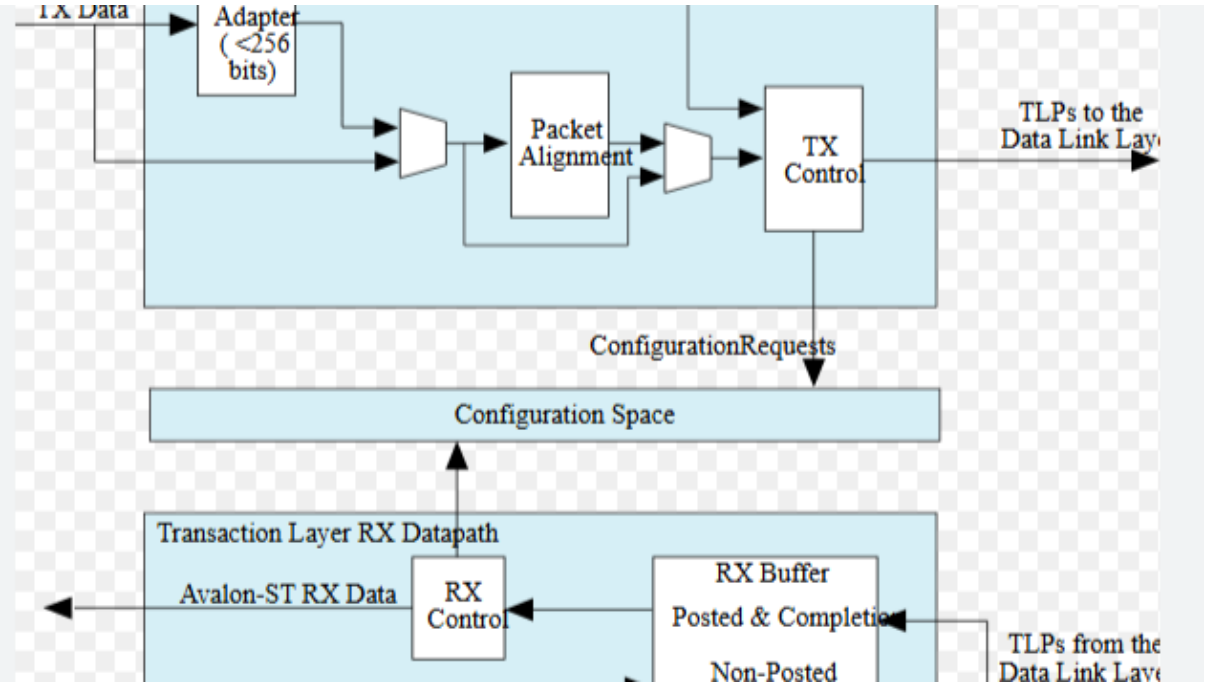
(optional check). Messages (e.g., Assert_INTX, Power Management, Error Signaling, Unlock, Set Power Slot Limit) not using the default traffic class.

- **Handling**: The Hard IP block deletes the malformed TLP and does not pass it to the Application Layer. As a fatal error, it can disrupt system stability and requires attention.

Key Takeaways

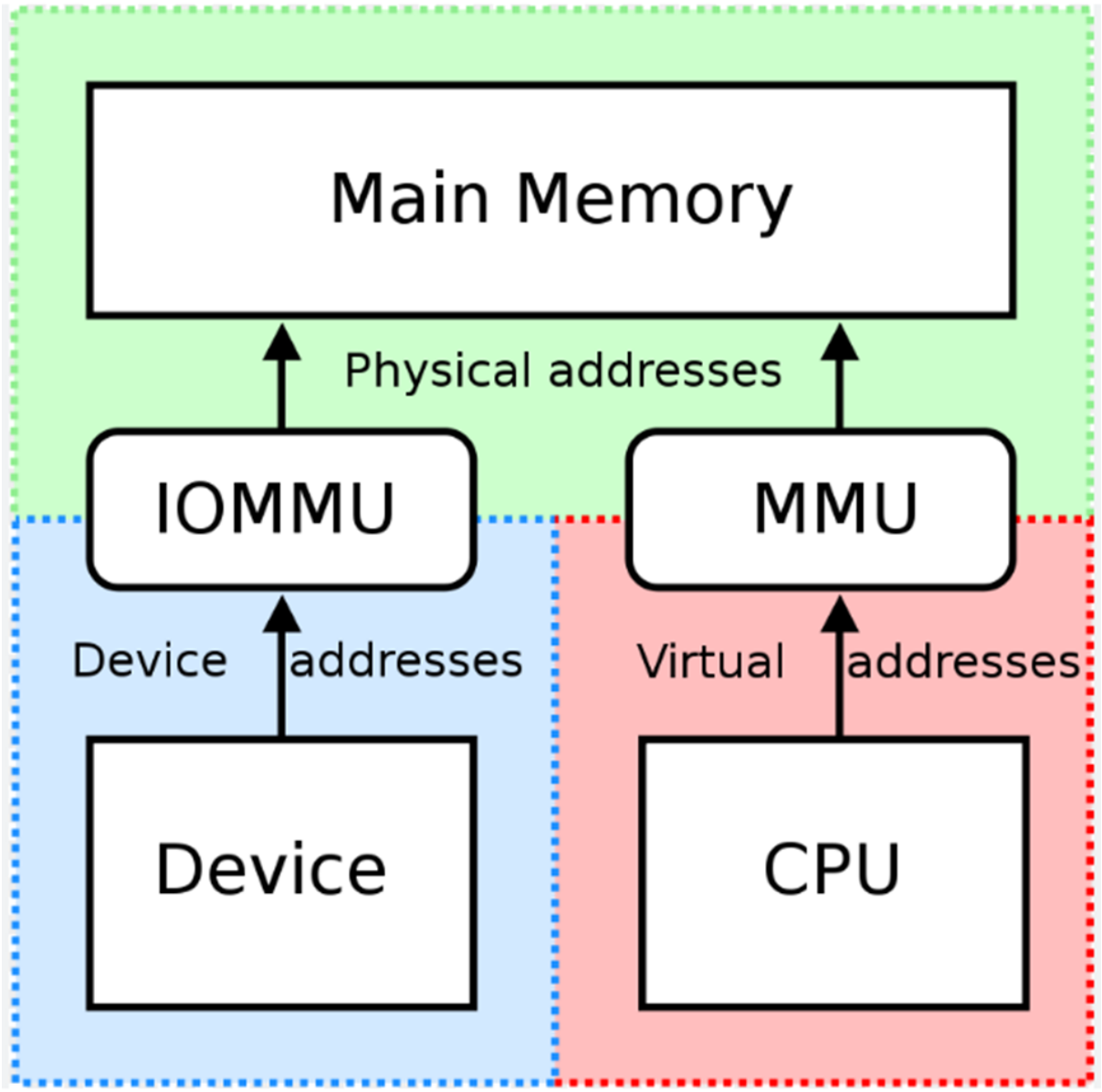
- **Non-fatal errors** (e.g., Poisoned TLP, ECRC Check Failed) typically allow the system to continue operating, with handling delegated to either the Hard IP block or Application Layer.
- **Fatal errors** (e.g., Receiver Overflow, FCPE, Malformed TLP) indicate severe issues that could compromise system integrity, often requiring higher-level intervention.
- The Hard IP block plays a critical role in filtering invalid TLPs, while the Application Layer handles specific cases like timeouts or poisoned data.

PCIe : Overview of Address Translation Service (ATS)



The **Address Translation Service (ATS)** is a key feature in the PCIe architecture designed to manage memory access for devices efficiently and securely. ATS ensures that different processes or applications running on a system operate within their own isolated memory spaces. It achieves this by translating **virtual addresses** (logical addresses used by programs or devices) into **physical addresses** (actual locations in memory). This translation mechanism is vital for maintaining system stability, preventing memory conflicts, and enhancing security.

In PCIe 6.0, ATS is enhanced with features like **Process Request Identifiers (PRIs)** and **Process Address Space Identifiers (PASIDs)**, which allow devices to better manage multiple processes. While these enhancements improve functionality, they also increase the complexity of testing and verification, requiring robust strategies to ensure reliability.



Introduction to Address Translation

Address translation is the process of converting a virtual address into a physical address. This is essential because programs and devices use virtual addresses to reference memory, which must be mapped to real memory locations. Two hardware units handle this translation:

- **Memory Management Unit (MMU)**: Translates virtual addresses from the CPU to physical addresses.
- **Input/Output Memory Management Unit (IOMMU)**: Translates virtual addresses from I/O devices (like PCIe devices) to physical addresses.

These units maintain mappings to ensure that each process or device accesses the correct memory without interfering with others.

Address Translation Process

The address translation process follows these steps:

1. A program or device references a memory location using a **virtual address**.
2. The virtual address is sent to the appropriate unit (MMU for the CPU, IOMMU for devices).
3. The unit checks its **Translation Lookaside Buffer (TLB)**, a cache storing recent translations.
4. If the virtual address is found in the TLB, the corresponding **physical address** is returned immediately.
5. If not found, the unit performs a **page table walk**—a lookup in memory tables—to determine the physical address.
6. The physical address is then returned to the processor or device for memory access.

This process ensures efficient and accurate memory access across the system.

Benefits of Address Translation

Address translation offers several critical advantages, particularly in modern and virtualized systems:

- **Memory Protection**: Prevents unauthorized memory access by isolating processes, enhancing system security.
- **Virtual Memory**: Enables the operating system to simulate more memory than physically available by using disk space as temporary storage, improving memory management.
- **Cache Coherency**: Ensures all copies of data in memory remain consistent, avoiding data discrepancies.
- **Device Isolation**: Keeps devices separate from each other, reducing conflicts and improving stability.
- **Reduced Processing Overhead**: By using an **Address Translation Cache (ATC)** in devices, ATS offloads translation tasks from the CPU, boosting performance.

These benefits are foundational to secure and efficient computing, especially in environments with multiple processes and devices.

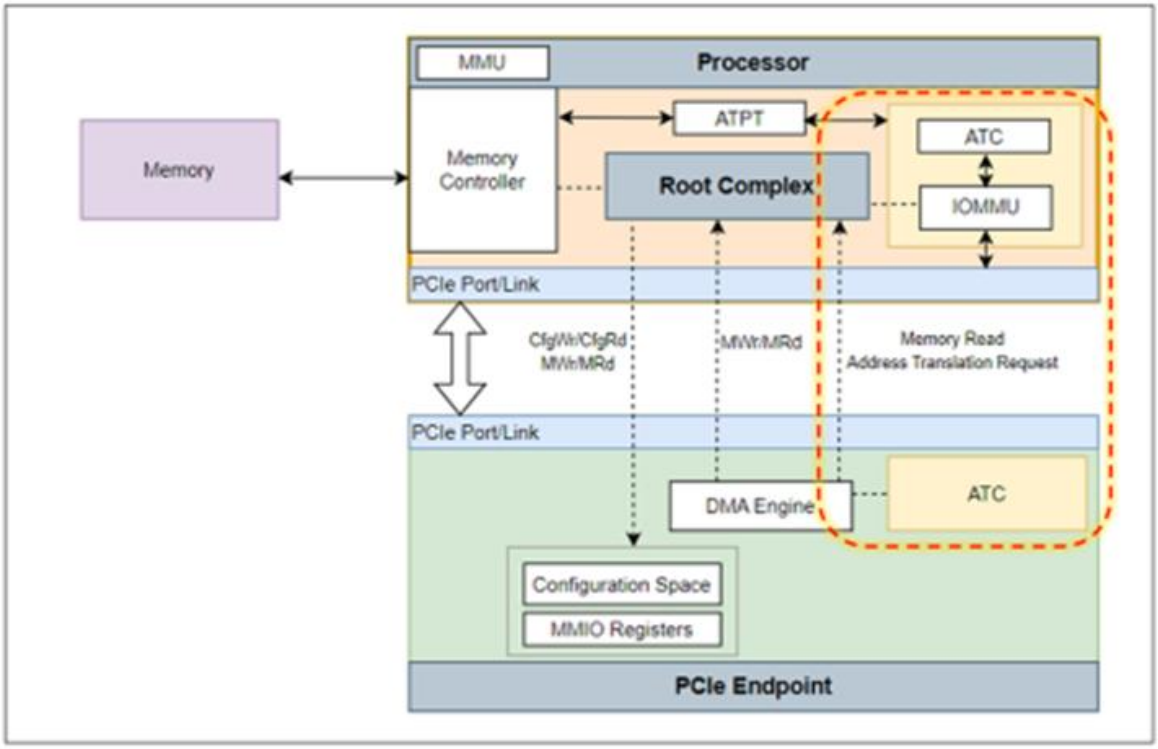
Key Components of ATS in PCIe 6.0

ATS in PCIe 6.0 relies on several components to function effectively:

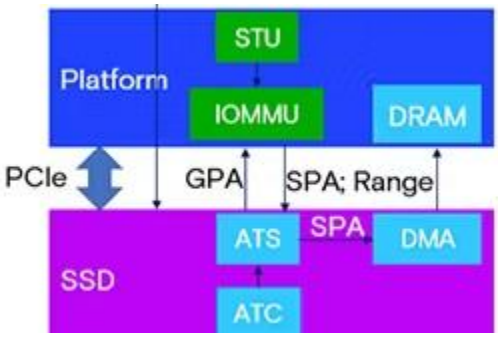
- **Translation Agent (TA)**: Typically an IOMMU or MMU, managed by software like a hypervisor. The TA translates addresses for PCIe devices, ensuring secure **Direct**

Memory Access (DMA) to host memory. It uses page tables and communicates with the **Root Complex (RC)** to confirm translation outcomes.

- **Address Translation Protection Table (ATPT)**: A data structure storing page tables with virtual-to-physical address mappings. The TA uses the ATPT to process PCIe requests, combining the **Routing ID** (identifying the device) and the address in the transaction.
- **Address Translation Cache (ATC)**: A cache within the PCIe device (specifically in Physical Functions) that stores recent address translations. The ATC speeds up memory access by reducing the need for repeated translations. It is populated through an **ATS Translation Request** sent by the device, followed by an **ATS Translation Completion** from the TA.



These components work together to enable devices to perform their own translations securely and efficiently.



How ATS Works in PCIe 6.0

ATS allows PCIe devices to handle address translations independently, rather than relying on the host CPU. Here's how it operates:

- When a device needs to access memory, it issues an **ATS Translation Request** with a virtual address.
- The TA processes this request using the ATPT and returns the physical address via an **ATS Translation Completion**.
- The device stores this mapping in its ATC, enabling fast subsequent memory accesses without CPU involvement.
- This hardware-based translation supports **DMA**, where devices directly read from or write to memory, bypassing the CPU.

This process reduces latency and frees the CPU for other tasks, enhancing overall system performance.

Summary and Benefits of ATS

ATS in PCIe 6.0 brings significant improvements to I/O device performance:

- **Reduced Latency**: Devices can access memory directly without waiting for CPU translations, speeding up operations.
- **Improved System Performance**: Offloading translations to devices reduces CPU workload, benefiting the entire system.
- **Efficient DMA**: Hardware-based translations enable faster and more secure memory access for PCIe devices.

These advantages are particularly valuable for applications requiring rapid memory access, such as high-performance computing or real-time systems.

Verification Strategy for ATS

The document highlights the importance of verifying ATS functionality due to its complexity, especially with PRIs and PASIDs. The verification process includes:

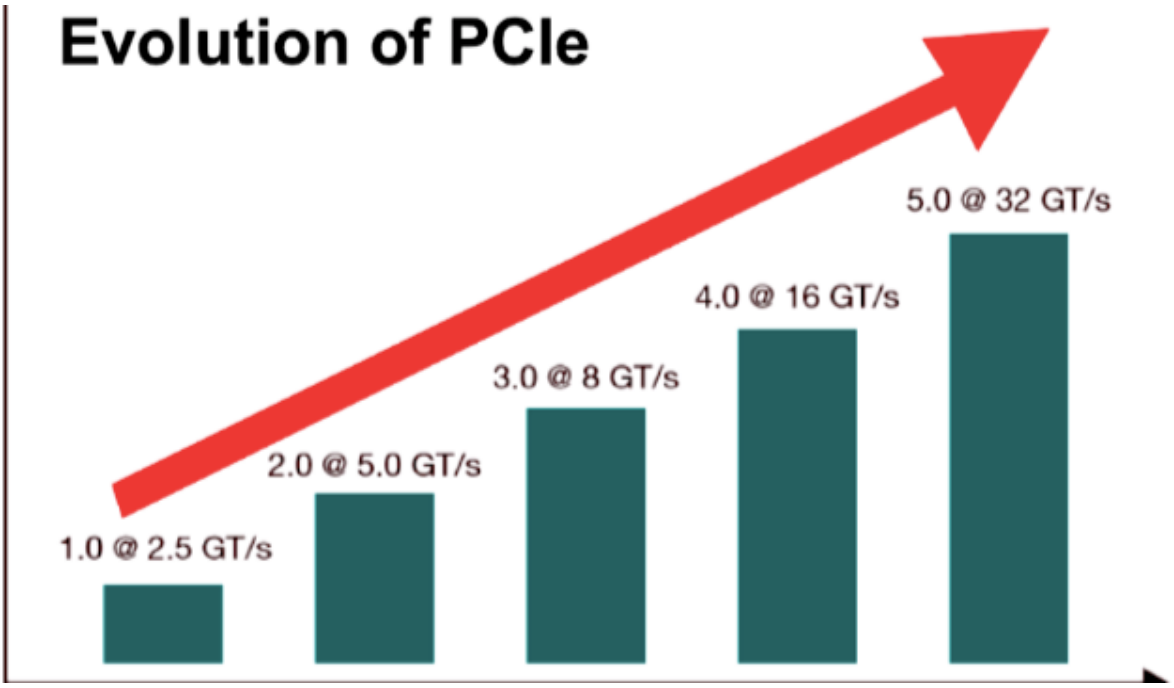
- **Functional Testing**: Ensures ATS works correctly across various scenarios.
- **Performance Testing**: Measures latency and throughput to confirm efficiency.
- **Stress Testing**: Pushes the system to its limits to identify weaknesses.

This systematic approach aims to catch issues early, ensuring ATS reliability in PCIe 6.0 environments.

Conclusion

The **Address Translation Service (ATS)** in PCIe 6.0 is a vital mechanism for managing memory access in modern systems. By translating virtual addresses to physical addresses, ATS ensures memory protection, enables virtual memory, maintains cache coherency, and isolates devices—all while reducing CPU overhead through hardware-based translations. With components like the TA, ATPT, and ATC, ATS enhances device performance and system efficiency, making it a cornerstone of PCIe 6.0.

PCIe Evolution, Questions and Answers



Evolution of PCIe Generations

The development of PCIe from its inception in 2003 to the latest generation, focusing on improvements in speed, bandwidth, encoding, and features.

Gen 1 (2003)

Key Features:

- Launched with a data rate of **2.5 GT/s** (Giga-transfers per second).
- Supported link widths: **x1, x2, x4, x8, x16**, a standard maintained across all generations.
- Used **8b/10b encoding**, which adds 25% overhead (2 bits per 8-bit data byte), yielding a total bandwidth of **4 Gb/s** for an x16 link.
- The encoding ensured DC balance and supported physical-layer functions like packet start/end markers.

Gen 2 (2007)

Key Improvements:

- Doubled the data rate to **5.0 GT/s**, increasing per-lane throughput from **250 Mbps** to **500 Mbps**.
- Retained **8b/10b encoding**.

- Introduced features for graphics applications and higher slot power limits for performance cards.
- **Backward Compatibility**: Gen 2 devices work with Gen 1 systems, reducing costs (e.g., 4 links at 5 GT/s match 8 links at 2.5 GT/s in bandwidth).

Gen 3 (2010)

Key Advancements:

- Increased data rate to **8.0 GT/s** (not 10 GT/s, to avoid design challenges like new PCB materials or shorter channel lengths).
- Switched to **128b/130b encoding**, reducing overhead to 1.5% and boosting bandwidth by 20%. Additional optimizations (e.g., removing K-codes) provided the remaining 20% to double bandwidth.
- Added features like I/O virtualization, device sharing, caching hints, and atomics for virtual machines and accelerators.

Gen 4 (2017)

Key Features:

- Doubled the data rate to **16.0 GT/s**, with a longer development time to ensure cost and power efficiency.
- Increased channel loss budget to **28 dB** (from 22 dB in Gen 3) and used advanced materials like Megatron-2.
- Platforms compensated for the delay by raising lane counts (up to 128 per CPU socket), tripling aggregate bandwidth.
- Enhanced RAS features for PCIe storage and deeper low-power states for mobile devices.

Gen 5 (2019)

Key Improvements:

- Doubled the data rate to **32.0 GT/s**, providing **32 Gbps** per lane and **128 Gb/s** for x16.
- Met demands of AI, ML, cloud computing, and 400G Ethernet with improved signal integrity (better traces, thicker PCBs).
- Supported alternate protocols (e.g., coherency) over PCIe pins.
- Channel loss extended to **36 dB**.

Gen 6 (2021)

Key Advancements:

- Doubled the data rate to **64.0 GT/s**.
- Adopted **PAM4 signaling** (4 voltage levels) to maintain channel reach, but introduced a higher bit error rate (BER).
- Added **Forward Error Correction (FEC)** and **FLIT encoding** (fixed-size packets) to manage errors efficiently with low latency.
- Maintained backward compatibility, ensuring coexistence with prior generations.

PCIe FAQs:

What are the key differences between a Root Complex, Endpoint, Switch, and Bridge in PCIe?

• **Root Complex (RC):** Connects the CPU to PCIe devices.

• **Endpoint (EP):** A device (GPU, SSD, NIC) that communicates with the Root Complex.

• **Switch:** Provides connectivity between multiple PCIe devices.

• **Bridge:** Connects legacy PCI/PCI-X devices to PCIe.

What is PCIe Enumeration?

PCIe enumeration is the process of detecting the devices connected to the PCIe bus.

As part of PCIe enumeration, switches and endpoint devices are allocated memory from the PCIe slave address space of the HOST.

The enumeration process includes: -

– Initialization of BAR address of endpoint and switches

– allocation and initialization of MSI/MSI-X address for the devices.

– empowers bus-mastering capabilities of the device to initiate transactions on the bus.

– Initialization of different capabilities of the devices like power-management, max-payload size, etc

How does PCIe address the challenge of clock skew in high-speed serial communication?

PCIe uses a reference clock at both ends of the link and incorporates clock data recovery mechanisms to address clock skew. This ensures that the receiving device can accurately recover the clock signal from the incoming data, even in the presence of variations in signal propagation times.

What are the layers of PCIe?

PCIe can be divided into three discrete logical layers: the Transaction Layer, the Data Link Layer, and the Physical Layer. Each of these layers is divided into two sections: one that processes outbound (to be transmitted) data and one that processes inbound (received) data.

What is PCIe Equalization?

When the PCIe link is at GEN3 or higher speeds, then there can be less signal quality (bad eye).

Equalization is the process of compensating for the distortion introduced by the channel. After passing through a band-limited channel, the high-frequency components of the signal are heavily attenuated, which distorts the signal and spreads it into subsequent symbol periods.

This is visible as a closed eye in the eye diagram. The process of equalization produces a sufficiently open eye, as in Figure 1, and decreases Inter Symbol Interference (ISI). This facilitates the easier recovery of transmitted symbols, ultimately reducing the Bit Error Rate (BER). The Link equalization procedure enables components to adjust the Transmitter and the Receiver setup of each lane to improve the signal quality (good eye). The equalization procedure can be initiated either autonomously or by software.

How does PCIe address the challenge of signal degradation over longer distances in high-speed serial links?

PCIe addresses signal degradation through the use of equalization techniques. Adaptive equalization adjusts the characteristics of the transmitted signal to compensate for signal degradation over longer distances, maintaining signal integrity and allowing for reliable communication.

What is trainable equalization?

Trainable equalization refers to the ability to change the tap coefficients. Each Tx, channel, and Rx combination will have a unique set of coefficients yielding an optimum signal-to-noise ratio. The training sequence comprises adjustments to the tap coefficients while applying a quality measurement to minimize the error.

What's the role of LTSSM (Link Training and Status State Machine) in PCIe?

LTSSM (**Link Training and Status State Machine**) manages the **link initialization, training, and recovery** in PCIe.

- It ensures **proper link negotiation** between devices.
- It transitions through states like **Detect, Polling, Configuration, Recovery, L0 (Active)**, etc.

- Responsible for **speed negotiation, lane alignment, and error recovery**.

Describe the concept of 'Replay Buffer' in PCIe. How does it improve data integrity?

Each transmitter stores recently sent TLPs in a **Replay Buffer**. If the receiver detects an error (via LCRC check) and sends a NAK, the transmitter **retransmits** the stored TLPs, ensuring data integrity.

What is the role of LCRC in PCIe? At which layer is it used?

LCRC (Link CRC) is a checksum added by the Data Link Layer to detect transmission errors. If an error is found, the receiver sends a **NAK** to request a retransmission.

Why does PCIe use credit-based flow control instead of traditional ACK/NACK?

Credit-based flow control **prevents buffer overflow** and **eliminates the need for per-packet ACKs**, reducing overhead.

What is the purpose of the PCIe configuration space.

The PCIe configuration space is a region of memory that contains configuration registers for each device on the bus. These registers store information about device capabilities, status, and other configuration details. Software can access this space to configure and query information about connected devices.

Discuss the concept of peer-to-peer communication in PCIe.

PCIe supports peer-to-peer communication, allowing devices to communicate directly without involving the CPU. This feature is particularly beneficial in scenarios where data needs to be transferred efficiently between devices, reducing latency and offloading the CPU from handling data transfers.

How does PCIe address the challenges of link latency, and what mechanisms are in place to optimize latency in the communication link?

PCIe minimizes link latency through features like TLP (Transaction Layer Packet) processing hints, which allow devices to provide information about the criticality of a transaction. Additionally, features such as split transactions and completion coalescing are employed to improve overall system responsiveness.

Discuss the impact of clock and power gating on power consumption in PCIe.

Clock and power gating are power management techniques used in PCIe to reduce power consumption during idle periods. By selectively turning off clock signals or powering down specific components when not in use, PCIe devices can achieve lower power states, contributing to overall energy efficiency in the system.

How does PCIe handle congestion?

If a receiver runs out of buffer space, it stops advertising credits, preventing further data transmission.

Discuss the role of the PCIe TLP prefix in optimizing data transfers.

The TLP (Transaction Layer Packet) prefix is a field in the TLP header that provides additional information about the payload. It includes information such as the payload type, allowing devices to optimize their handling of the data. This feature helps enhance the efficiency of data transfers in PCIe.

How does PCIe handle interrupt delivery?

PCIe uses Message Signaled Interrupts (MSI) and MSI-X to handle interrupts. Instead of sharing a limited number of interrupt lines as in traditional PCI, each device generates its own interrupt request by sending a message to the interrupt controller.

Explain the role of the PCIe Extended Capability Structure in enhancing device capabilities.

The PCIe Extended Capability Structure is a mechanism that allows devices to expose additional capabilities beyond the standard PCIe configuration space. This enables devices to provide more detailed information about their features and functionalities, enhancing interoperability and facilitating advanced configuration.

What is the role of the PCIe retimer, and how does it improve signal integrity in the communication link?

A PCIe retimer is a component that helps restore and reshape signals in the communication link, improving signal integrity. It is often used in scenarios where signal degradation occurs due to factors like long trace lengths or the presence of connectors. The retimer ensures that the received signals are of sufficient quality for reliable communication.

Explain the concept of ordered sets in PCIe and their significance in link training.

Ordered sets are special sequences of bits used for link training and maintenance. They help establish and maintain synchronization between transmitting and receiving devices. Ordered sets contain specific bit patterns that indicate different link states, helping ensure reliable and error-free communication.

What is scrambling?

PCI Express utilizes data scrambling to diminish the chance of electrical resonances on the link. PCI Express specification defines a scrambling/descrambling algorithm that is carried out utilizing a linear feedback shift register.

Scrambling is a technique where a realized binary polynomial is applied to a data stream in a feedback topology. Since the scrambling polynomial is known, the data can be recovered by running it through a feedback topology using the inverse polynomial.

PCI Express accomplishes scrambling or descrambling by performing a serial XOR operation to the data with the seed output of a Linear Feedback Shift Register (LFSR) synchronized between PCI Express devices.

How does Flow Control work in PCIe?

Flow Control in PCIe prevents buffer overflow and ensures smooth data transmission. It operates using:

Credit-Based Flow Control: Transmitters request permission before sending packets.

Receivers allocate **credits** for different buffer types (Posted, Non-Posted, Completion).

Data is sent only when sufficient credits are available, preventing congestion.

Flow control ensures **reliable data transfer**, avoiding **packet drops** and **deadlocks**.

Can you explain the concept of virtual channels in PCIe?

Virtual channels in PCIe allow the division of the physical link into multiple virtual lanes, each with its own flow control mechanism. This feature helps prioritize and manage traffic based on different Quality of Service (QoS) requirements, enhancing the overall efficiency of data transfer.

How does PCIe handle QoS (Quality of Service)?

PCIe uses **Traffic Classes (TC) and Virtual Channels (VC)** to ensure high-priority data gets transmitted first.

Explain how PCIe performs error detection and correction. What are the different types of errors?

PCIe uses:

- **LCRC (Link CRC):** Detects transmission errors.
- **ACK/NAK Mechanism:** Retransmits corrupted data.
- **Advanced Error Reporting (AER):** Reports and handles errors at the OS level.

Types of errors:

- **Correctable Errors:** Automatically fixed (e.g., bit flips).
- **Uncorrectable Errors:** Fatal errors requiring system intervention.

How does PCIe support error reporting and recovery in the event of a link failure?

PCIe incorporates Advanced Error Reporting (AER) mechanisms to detect, report, and recover from errors in the communication link. When errors occur, devices can generate

error messages, allowing the system to take appropriate actions, such as retraining the link or isolating the affected component.

What is a Non-Posted vs. Posted transaction in PCIe?

• **Posted Transaction (P):** No response required (e.g., Memory Write).

• **Non-Posted Transaction (NP):** Requires a response (e.g., Memory Read).

How does PCIe support atomic operations?

PCIe supports **AtomicOps** (e.g., Atomic Compare & Swap, Atomic Fetch & Add) to ensure **thread-safe** operations across multiple devices.

Explain the difference between DLLP and TLP in PCIe.

Feature	TLP (Transaction Layer Packet)	DLLP (Data Link Layer Packet)
Purpose	Carries actual transaction data (Read/Write Requests, Config, etc.)	Ensures link reliability (ACK/NAK, Flow Control, Power Management)
Layer	Transaction Layer	Data Link Layer
Size	Variable	Small, typically 8 bytes
Examples	Memory Read/Write, I/O transactions, Configuration packets	ACK/NAK for error handling, Flow Control Updates, Power Management messages

Explain the concept of Max Payload Size (MPS) and Max Read Request Size (MRRS).

• **MPS (Max Payload Size):** Defines the largest TLP payload a device can send.

• **MRRS (Max Read Request Size):** Limits the maximum data a single read request can fetch.

Larger values improve throughput but may increase latency.

What are the different power management states in PCIe, and how do they contribute to energy efficiency?

PCIe supports various power management states, including L0 (fully operational), L0s (low-power idle), L1 (lower-power idle), and L2 (power-off). Devices can transition between these states based on workload requirements, contributing to overall energy efficiency by dynamically adjusting power

consumption.

How does PCIe handle latency-sensitive data, such as real-time audio/video streams?

PCIe supports **Traffic Classes (TC) and Virtual Channels (VC)** to prioritize latency-sensitive transactions.

What is the function of a Completion TLP?

A **Completion TLP (Cpl)** is used to return data from a **Non-Posted Request** (e.g., memory read response).

Discuss the concept of "completion timeout" in PCIe.

Completion timeout is the maximum time a device should take to respond to a transaction request. If a device doesn't respond within this time, it is considered a timeout, and the link may be reset. This mechanism ensures that devices operate within specified time limits, preventing system hangs.

What is the role of the PCIe bifurcation feature, and how does it impact system design?

PCIe bifurcation allows a single physical slot to be split into multiple logical slots, each with its own set of lanes. This feature is valuable in optimizing the use of available PCIe lanes, especially in systems where the number of physical slots is limited. It offers flexibility in accommodating different device configurations.

Explain the concept of ATS (Address Translation Services) in PCIe.

Address Translation Services (ATS) in PCIe enables a device to request translation services from the endpoint that manages its address space. This is particularly useful in virtualized environments where devices might have different address spaces, and ATS facilitates efficient address translation without involving the CPU.

Explain the concept of hot swapping in PCIe.

Hot swapping refers to the ability to replace or add PCIe devices while the system is running. PCIe supports hot swapping through features like surprise removal, where the system can detect the removal or addition of a device and dynamically reconfigure the PCIe topology without requiring a system reboot.

How does PCIe support hot-plugging of devices?

PCIe supports hot-plugging by utilizing the Advanced Error Reporting (AER) mechanism. When a device is hot-plugged, the AER capability allows the system to detect the change and reconfigure the PCIe topology without requiring a system reboot. This feature is particularly useful for server environments where uninterrupted operation is critical.

Explain how PCIe supports multi-function devices and the advantages of this capability.

PCIe allows a single physical device to have multiple logical functions, each with its own set of configuration space and resources. This enables more efficient use of PCIe slots and resources, as a single physical device can provide multiple functionalities without the need for separate physical slots.

What is Forward Error Correction (FEC), and how is it utilized in the PCIe 6.0 specification?

Lightweight Forward Error Correction (FEC) and Strong Cyclic Redundancy Check (CRC) are the two essential techniques utilized in the PCIe 6.0 specification to address errors.

With the 64 GT/s data rate enabled by PAM4 encoding in the PCIe 6.0 specification, the bit error rate (BER) was several orders of magnitude higher than the 10-12 BER in all prior generations. FEC and CRC mitigate the bit error rate and allow the PCIe 6.0 specification to reach new performance degrees.

Flit Mode supports the higher BER expected in PAM4 (10-6 vs. 10-12 in NRZ). This can provide increased resilience in NRZ environments.

How does PCIe implement ordering rules? Explain the concept of relaxed ordering.

PCIe normally maintains strict ordering for transactions, but **Relaxed Ordering (RO)** allows certain transactions (e.g., writes) to bypass others for improved performance.

What software changes were needed to take advantage of Flit Mode?

Much care has been taken to keep away from significant impacts to existing software. However, some changes could not be avoided in order to take full advantage of Flit mode. Here are some examples:

- The new TLP format changes how we interpret error logs.
- Inter-hierarchy (a.k.a., inter-segment) routing allows the use of larger trees of PCIe technology devices to communicate directly.
- Defined routing behavior for all TLP-type codes allows for additional TLPs to be defined without needing to change switches.

Discuss the role of ECN (Explicit Congestion Notification) in PCIe.

Explicit Congestion Notification in PCIe allows devices to communicate congestion information, enabling more efficient traffic management. Devices can signal congestion explicitly, allowing the system to take appropriate actions such as re-routing traffic or adjusting transmission rates to alleviate congestion.

Does putting a PCIe Gen3 video card in a Gen4 slot improve performance?

No, on the off chance that the designs card itself is PCIe 3.0, placing it in a faster 4.0 slot won’t give any advantage since they will be working at Gen3 speed.

What’s the role of Forward Error Correction (FEC) in Gen6, and how does it impact performance?

Role of FEC in PCIe Gen6:

• FEC (Forward Error Correction) is introduced to correct transmission errors introduced by PAM4 signaling.

• Since PAM4 has a higher Bit Error Rate (BER) than NRZ, FEC helps ensure data reliability.

Impact on Performance:

Error Correction: Reduces the need for retransmissions, improving overall efficiency.

Increased Latency: FEC adds a small processing delay (~2-4ns), but the benefits of higher bandwidth outweigh this.

Flit Mode Integration: PCIe Gen6 operates in 256-byte flits (Flow Control Units), where FEC is embedded for error handling.

What are the key differences between PCIe and NVMe, and how do they complement each other in modern storage architectures?

PCIe is a high-speed interconnect standard, while NVMe (Non-Volatile Memory Express) is a protocol designed specifically for storage devices, often using PCIe as the physical interface. PCIe provides the high-speed communication channel, and NVMe optimizes the communication protocol for low-latency access to non-volatile storage, resulting in faster storage solutions.

Why did PCIe Gen6 switch to PAM4 instead of NRZ?

· PCIe Gen6 adopted PAM4 (Pulse Amplitude Modulation - 4 levels) instead of NRZ (Non-Return-to-Zero) because:

· Higher Data Rate at the Same Frequency: PAM4 packs 2 bits per clock cycle instead of 1 (as in NRZ), doubling bandwidth without increasing clock speed.

· Signal Integrity Concerns: At 64 GT/s, using NRZ would have caused excessive signal loss and increased power consumption.

· Power Efficiency: PAM4 enables lower power per bit transmission, improving overall efficiency.

· Challenges with PAM4: It has higher bit error rates (BER), which is why Forward Error Correction (FEC) is introduced in Gen6.

Trade-off:

PAM4 is more complex and requires additional error correction (FEC), but it allows PCIe to scale without significantly increasing power consumption.

How does PCIe ensure security in data transmission?

PCIe incorporates security features such as ECN (End-to-End data Corruption Notification) and TLP processing hints to enhance data integrity and security. Additionally, features like Access Control Services (ACS) help isolate and protect devices from unauthorized access, contributing to a more secure PCIe ecosystem.

What is ACS.

Disbale Peer-to-Peer PCIe transaction

Access Control Services (ACS) provides a mechanism by which a Peer-to-Peer PCIe transaction can be forced to go up through the PCIe Root Complex. ACS can be thought of as a kind of gate-keeper —preventing unauthorized transactions from occurring.

Without ACS, it is possible for a PCIe Endpoint to either accidentally or intentionally (maliciously) write to an invalid/illegal area on a peer endpoint, potentially causing problems.